ICN 2016

The Fifteenth International Conference on Networks

ISBN: 978-1-61208-450-3

SOFTNETWORKING 2016

The International Symposium on Advances in Software Defined Networks

February 21 - 25, 2016

Lisbon, Portugal

ICN 2016 Editors

Eugen Borcoci, University Politehnica of Bucharest, Romania

Pascal Lorenz, University of Haute Alsace, France

Tibor Gyires, Illinois State University, USA

Carlos Becker Westphall, University of Santa Catarina, Brazil

Andy Snow, Ohio University, USA

ICN 2016

Forward

The Fifteenth International Conference on Networks (ICN 2016), held between February 21-25, 2016 in Lisbon, Portugal continued a series of events focusing on the advances in the field of networks.

ICN 2016 welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard fora or in industry consortia, survey papers addressing the key problems and solutions, short papers on work in progress, and panel proposals.

The conference had the following tracks:

- Performance evaluation, tools, simulation
- Communications security, reliability, availability
- Software Defined Networks and Network Functions Virtualization
- Internet of Things, cellular cognitive networks

The conference also featured the following symposium:

- **SOFTNETWORKING 2016*, The International Symposium on Advances in Software Defined Networking and Network Functions Virtualization***

We take here the opportunity to warmly thank all the members of the ICN 2016 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to ICN 2016. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the ICN 2016 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope ICN 2016 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the field of networks. We also hope that Lisbon, Portugal, provided a pleasant environment during the conference and everyone saved some time to enjoy the beauty of the city.

ICN 2016 Advisory Committee

Pascal Lorenz, University of Haute Alsace, France
Tibor Gyires, Illinois State University, USA
Carlos Becker Westphall, University of Santa Catarina, Brazil
Iwona Pozniak-Koszalka, Wroclaw University of Technology, Poland

SOFTNETWORKING 2016 Advisory Committee

Eugen Borcoci, University Politehnica of Bucharest, Romania
Pedro A. Aranda Gutiérrez, Telefónica, Spain
Nicola Ciulli, Nextworks, Italy
Marc Sune, Berlin Institute for Software Defined Networks GmbH, Germany
Wolfgang John, Ericsson Research, Sweden

ICN 2016

Committee

ICN 2016 Advisory Committee

Pascal Lorenz, University of Haute Alsace, France
Tibor Gyires, Illinois State University, USA
Carlos Becker Westphall, University of Santa Catarina, Brazil
Iwona Pozniak-Koszalka, Wroclaw University of Technology, Poland

ICN 2016 Technical Program Committee

Alireza Abdollahpouri, University of Kurdistan - Sanandaj, Iran
Colin Allison, University of St Andrews, UK
Natalia Amelina, St. Petersburg State University, Russia | Norwegian University of Science and Technology, Norway
Max Agueh, LACSC - ECE Paris, France
Kari Aho, University of Jyväskylä, Finland
Pascal Anelli, Université de la Réunion, France
Cristian Anghel, Politehnica University of Bucharest, Romania
Mohammad Ashiqur Rahman, Tennessee Tech University, USA
Jocelyn Aubert, Luxembourg Institute of Science and Technology (LIST), Luxembourg
Harald Baier, Hochschule Darmstadt, Germany
Katherine Barabash, IBM Research - Haifa, Israel
Alvaro Barradas, University of Algarve, Portugal
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Zdenek Becvar, Czech Technical University in Prague, Czech Republic
Djamel Benferhat, University of South Brittany, France
Ilham Benyahia, Université du Québec en Outaouais - Gatineau, Canada
Robert Bestak, Czech Technical University in Prague, Czech Republic
Jun Bi, Tsinghua University, China
Bruno Bogaz Zarpelão, State University of Londrina (UEL), Brazil
Patrick-Benjamin Bök, Weidmueller Group, Germany
Fernando Boronat Seguí, Universidad Politécnica de Valencia, Spain
Radoslav Bortel, Czech Technical University, Czech Republic
Agnieszka Brachman, Silesian University of Technology - Gliwice, Poland
Arslan Broemme, GI BIOSIG - GI e.V., Germany
Matthias R. Brust, Technological Institute of Aeronautics, Brazil
Damian Bulira, Wroclaw University of Technology, Poland
Manoel Camillo Penna, Pontifícia Universidade Católica do Paraná, Brazil

Pascal Lorenz, University of Haute Alsace, France
Richard Lorion Université de la Réunion, France
Damien Magoni, University of Bordeaux, France
Ahmed Mahdy, Texas A&M University - Corpus Christi, USA
Lefteris Mamatas, University of Macedonia, Greece
Zoubir Mammeri, IRIT - Paul Sabatier University - Toulouse, France
Anna Manolova Fagertun, Technical University of Denmark, Denmark
Gustavo Marfia, University of Bologna, Italy
Rui Marinheiro, ISCTE - Lisbon University Institute, Portugal
Antonio Martín, Seville University, Spain
Rafael Mendes, Federal University of Santa Catarina, Brazil
Boris M. Miller, Monash University/ Institute for Information Transmission Problems, Australia
Pascale Minet, INRIA - Rocquencourt, France
Mohamed Mohamed, IBM US Almaden, USA
Mario Montagud Climent, Universidad Politécnica de Valencia, Spain
Katsuhiro Naito, Aichi Institute of Technology, Japan
Go-Hasegawa, Osaka University, Japan
Constantin Paleologu, University Politehnica of Bucharest, Romania
Konstantinos Patsakis, University of Piraeus, Greece
João Paulo Pereira, Polytechnic Institute of Bragança, Portugal
Kun Peng, Institute for Infocomm Research, Singapore
Dirk Pesch, Nimbus Centre for Embedded Systems Research - Cork Institute of Technology, Ireland
Ionut Pirnog, "Politehnica" University of Bucharest, Romania
Marcial Porto Fernandez, Universidade Estadual do Ceara (UECE), Brazil
Iwona Pozniak-Koszalka, Wroclaw University of Technology, Poland
Jani Puttonen, Magister Solutions Ltd., Finland
Mahshid Rahnamay-Naeini, Texas Tech University, USA
Shankar Raman, Indian Institute of Technology - Madras, India
Victor Ramos, UAM-Iztapalapa, Mexico
Priyanka Rawat, INRIA Lille - Nord Europe, France
Shukor Razak, Universiti Teknologi Malaysia (UTM), Malaysia
Yenumula B. Reddy, Grambling State University, USA
Eric Renault, Institut Mines-Télécom - Télécom SudParis, France
M. Elena Renda, IIT - CNR, Italy
Krisakorn Rerkrai, RWTH Aachen University, Germany
Karim Mohammed Rezaul, Glyndwr University - Wrexham, UK
Wouter Rogiest, Ghent University, Belgium
Simon Pietro Romano, University of Napoli Federico II, Italy
Angelos Rouskas, University of Piraeus, Greece
Jorge Sá Silva, University of Coimbra, Portugal
Teerapat Sanguankotchakorn, Asian Institute of Technology - Klong Luang, Thailand
Susana Sargento, University of Aveiro, Portugal
Panagiotis Sarigiannidis , University of Western Macedonia - Kozani, Greece

Masahiro Sasabe, Graduate School of Information Science - Nara Institute of Science and Technology, Japan
Thomas C. Schmidt, HAW Hamburg, Germany
Hans Scholten, University of Twente- Enschede, The Netherlands
Dimitrios Serpanos, ISI/RC Athena & University of Patras, Greece
Narasimha K. Shashidhar, Sam Houston State University - Huntsville, USA
Pengbo Si, Beijing University of Technology, P.R. China
Frank Siqueira, Federal University of Santa Catarina - Florianopolis, Brazil
Peter Skworcow, MontFort University - Leicester, UK
Karel Slavicek, Masaryk University Brno, Czech Republic
Andrew Snow, Ohio University, USA
Arun Somani, Iowa State University - Ames, USA
Kostas Stamos, University of Patras, Greece
Lars Strand, Nofas Management, Norway
Miroslav Sveda, Brno University of Technology, Czech Republic
Maciej Szostak, Wroclaw University of Technology, Poland
Nabil Tabbane, SUPCOM, Tunisia
János Tapolcai, Budapest University of Technology and Economics, Hungary
Carlos Miguel Tavares Calafate, Universidad Politécnica de Valencia, Spain
Ken Turner, The University of Stirling, UK
Emmanouel Varvarigos, University of Patras, Greece
Dario Vieira, EFREI, France
Calin Vladeanu, University Politehnica of Bucharest, Romania
Matthias Vodel, Technische Universitaet Chemnitz, Germany
Lukas Vojtech, Czech Technical University in Prague, Czech Republic
Krzysztof Walkowiak, Wroclaw University of Technology, Poland
Boyang Wang, Xidian University, China
Tingka Wang, London Metropolitan University, UK
You-Chiun Wang, National Sun Yat-sen University, Taiwan
Yufeng Wang, University of South Florida - Tampa | NEC-Labs America - Princeton, USA
Gary Weckman, Ohio University, USA
Alexander Wijesinha, Towson University, USA
Maarten Wijnants, Hasselt University-Diepenbeek, Belgium
Bernd Wolfinger, University of Hamburg, Germany
Kok-Seng Wong, SoongSil University, South Korea
Chandrika Worrall, Alcatel-Lucent Telecom Ltd, UK
Qin Xin, University of the Faroe Islands, Faroe Islands
Lei Xiong, National University of Defense Technology - ChangSha, China
Qimin Yang, Harvey Mudd College-Claremont, USA
Vladimir Zaborovski, Polytechnic University of Saint Petersburg, Russia
Pavel Zahradnik , Czech Technical University Prague, Czech Republic
Morteza Mohammadi Zanjireh, Glasgow Caledonian University, UK
Arkady Zaslavsky, CSIRO ICT Centre & Australian National University - Acton, Australia
Sherali Zeadally, University of Kentucky, USA

Bing Zhang, National Institute of Information and Communications Technology - Yokosuka, Japan
Bo Zhao, Samsung Research America, USA
Tayeb Znati, University of Pittsburgh, USA
André Zúquete, IEETA - University of Aveiro, Portugal

SOFTNETWORKING 2016

SOFTNETWORKING 2016 Advisory Committee

Eugen Borcoci, University Politehnica of Bucharest, Romania
Pedro A. Aranda Gutiérrez, Telefónica, Spain
Nicola Ciulli, Nextworks, Italy
Marc Sune, Berlin Institute for Software Defined Networks GmbH, Germany
Wolfgang John, Ericsson Research, Sweden

SOFTNETWORKING 2016 Technical Program Committee

Osianoh Glenn Aliu, Fraunhofer FOKUS - NETwork Research, Germany
Pedro A. Aranda Gutiérrez, Telefónica, Spain
Radu Badea, University Politehnica of Bucharest, Romania
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Robert Bestak, Czech Technical University in Prague, Czech Republic
Eugen Borcoci, University Politehnica of Bucharest, Romania
Nicola Ciulli, Nextworks, Italy
Didier Colle, iMinds - Ghent University, Belgium
Daniel Corujo, University of Aveiro, Portugal
Paolo Crosta, ITALTEL SpA, Italy
Christian Esteve Rothenberg, University of Campinas, Brazil
Roberto Gonzalez, NEC Laboratories Europe, Germany
Xavier Hesselbach-Serra, Universitat Politècnica de Catalunya (UPC) - Barcelona, Spain
Wolfgang John, Ericsson Research, Sweden
Eiji Kawai, National Institute of Information and Communications Technology, Japan
Wolfgang Kiess, DOCOMO Euro-Labs, Germany
Zoltán Lajos Kis, Ericsson, Hungary
Diego Kreutz, University of Luxembourg, Luxembourg
Maciej Kuzniar, EPFL, Switzerland
Francesco Longo, Università degli Studi di Messina, Italy
Radu Lupu, University Politehnica of Bucharest, Romania
Lefteris Mamatas, University of Macedonia, Greece
Ioannis Moscholios, University of Peloponnese, Greece
Milosz Przywecki, Poznan Supercomputing and Networking Center, Poland
Fernando Ramos, University of Lisbon, Portugal
Yuji Sekiya, University of Tokyo, Japan

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Toward a Framework for VM organisation based on Multi-Objectives

Guilherme Arthur Geronimo
Federal University of Santa Catarina
Florianópolis, Brazil
guilherme.geronimo@ufsc.br

Rafael Brundo Uriarte
IMT Institute for Advanced Studies
Lucca, Italy
rafael.uriarte@imtlucca.it

Carlos Becker Westphall
Federal University of Santa Catarina
Florianópolis, Brazil
carlos.westphall@ufsc.br

***Abstract*—This paper proposes a flexible framework to improve the quality of Virtual Machine's placements, in Clouds. It organises them by relocating the VMs based on the Multiple-Objectives of the environment. These Objectives are represented by Rules, Qualifiers, and Costs, which can be extended and prioritised. Based on Evolutionary Searches, the framework theoretically guarantees the adoption of a better set of Placements. More specifically, it seeks the non-dominated solutions (Pareto's Dominance concept) and compares then considering the implementation cost of the scenario and its benefits. In contrast to existing solutions that address specific objectives, our framework was devised to support many types of objectives and to be easily extensible, which enables the implementation of new and generic prioritised elements. Moreover, we conducted experiments using data from a real Cloud environment and show the flexibility of our approach and its scalability.**

Keywords*—*Virtual Machine Placement; Cloud Computing; Multi-Objective Optimisation.

I. INTRODUCTION

Although Cloud Computing (CC) can bring many benefits to consumers and providers, Cloud's mismanagement usually accentuates problems related to waste of resources. For instance, performance degradation due to noisy neighbours rise of thermal hotspots on data centre and shortage of resources due constant migrations [1][2].

Many approaches were proposed to mitigate this problem, such as Simulation-Based, Policy-Based, Bin Packing and Evolutionary Algorithms [3]-[5]. However, these proposals usually focus on specific objectives, e.g., on decreasing the energy consumption. This limitation hinders the adoption of these approaches, since Cloud's objectives, policies and priorities vary, and these proposals cannot adapt to these needs.

To address this limitation some works, e.g., [6]-[8], consider Virtual Machine Placement (VMP) problem as a multiple objective (MO) optimisation problem, that, simultaneously, tries to minimise the total resource wastage, power consumption and thermal dissipation costs. This wider view enables managers to consider other facets of VMP, such as internal policies and Service Level Objectives (SLO).

Due to those many facets, to solve MOs conflicts and guarantee a fast convergence, proposals usually adopt Evolutionary strategies that build non-dominated scenarios sets (using the Pareto Dominance concept). In this case, a non-dominated scenario is better in at least one objective and, at the same time, not worse in any other objective, compared to a base scenario. Despite the efficiency, this strategy stagnates in environments with many objectives, i.e., possibly reaching a state where none result is good enough for all objectives simultaneously. Moreover, the selection of the best scenario is also a challenge due the limited evaluation's strategies, their execution time and cost. Besides, none of the existing solutions consider, at the same time, the important Cloud's characteristics: SLO, SLA, policies and costs.

Consequently, these aspects need to be represent and implement in a standard manner, a model to solve VMP problems. Despite the fact of many works address this issue, none of them aimed at comprehend agnostic-objective methods, focusing in specific and limited objectives. Thus, to the best of our knowledge, no suitable model meets our needs to represent different types of evaluations, quantifiable and qualifiable.

Nonetheless, the VMP problem can be divided in sub-issues, such as Provisioning, Organisation and Dynamic Allocation of VMs. Even though, the environment objectives are always the same, regardless which sub-issue are being solved. However, the majority of the proposals treat them separately and differently – in terms of objectives and strategies.

Considering these limitations, we designed an easily extensible framework to address the VMP organisation problem. This framework adopts a flexible approach that enables the assessment and comparison of a single placements to the whole clusters, enabling evaluations with grater precision. Moreover, it uses MO qualification functions to provide a placement to new Virtual Machines (VMs) or select and relocate VMs to non-dominated scenarios. After filtering the possible scenarios, it chooses the best result regarding its benefits and implementation costs.

This framework takes advantage of constraints and SLAs to reduce the computation cost, enabling a fast local-optimal search. The main contributions of this framework are: (i) the support to generic multiple objectives and (ii) objective prioritisation. The rest of this paper is organised as follows: Section II provides the background concepts and the related works; Section III define the Cloud model; Section IV presents the methods and their designs; Section V describes the framework's tests, implementation and results; Section VI concludes the article addressing the future works and open issues.

 ISBN: 978-1-61208-450-3

II. Background and Related Works

The first part of this section presents important concepts used to discuss the related works in the second section.

A. Background

Extract and Relocate is an approach to select and migrate VMs (one or more) and is employed to improve the overall state of a cloud. The selection of VMs is commonly performed using heuristics or triggers, which normally attempt to choose the special VMs, according to some criteria, and migrate these.

The Pareto's Dominance concept can be used to filter the many possibilities of Relocation. The Pareto Set is a set of non-dominated solutions, i.e., scenarios that are better in at least one objective and, at the same time, not worse in any other objective compared to a base scenario [9]. Let $S_1 = (p_1, ..., p_y)$ and $S_2 = (q_1, ..., q_y)$ be two scenarios with y placements evaluations, S_1 is said to *dominate* S_2 if $p_i \geq q_i$ for all $i = 1, ..., y$ and $S_1 \neq S_2$. Figure 1 graphically represents this evaluation considering two objectives. The scenarios marked with a $(\star)$ are the non-dominated solutions and the $(\otimes)$ are dominated by their successors.

In order to build the set of possible scenarios, the *MO Variable Neighbourhood Search* [10] uses the concepts of *Solution Dominance* to explore the neighbour possibilities. Thus, instead of searching all possible scenarios, the approach explores only the non-dominated ones, building a Pareto Set with the dominant solutions found. However, exploring only the non-dominated solutions restricts searching space. Therefore, they vary the neighbourhoods restarting the search from different random scenarios. The union of the Pareto's Sets forms the Pareto Front set, i.e., all the $(\star)$ in Figure 1.

Furthermore, the selection of a scenario from the Pareto Front is not a deterministic choice, depending on the problem's nature and the applied model. There are a wide variety of approaches that could be adopted for VMP, for instance, Zitzler et al. [11] compared twenty-two generic assessments that could be used in MOPs, such as number of fulfilled goals, distance from base scenario and error ratio.

Despite of the variety, those indicators do not consider the cost to implement a scenario, which is an important decision aspect in VMP. Usually, the *Costs* are treated as a consequence of energy consumption or as another assessment to be minimised [5][12][13]. However, Cost is any commodity that Cloud Providers are willing to spend in order to achieve an amount of benefit. Its nature is a variable aspect in kind – such as time, computational resources or money – and it should be part of the decision process.

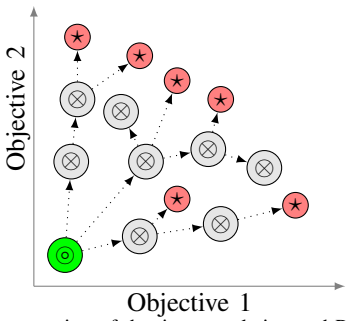


Figure 1. Representation of dominance relation and Pareto Frontier

B. Related Works

Most of the works regarding VMP focus on specific issues, such as a set of services [13], for network optimisation [14][15] and energy consumption [4][16][17]. Such methods have limited applicability and, therefore, are not considered in this work.

Nevertheless, some proposals adopt approaches which could be adapted and improved to manage generic MOs. For instance, the following proposals adopt the *Extract and Relocate* approach to solve their specific VMP problems. Beloglazov et al. [4] proposes four heuristics to select a VM to be extracted based on the VM's load, the number of migrations, growth potential and random choice. Then, relocate them using a variation of Best Fit Decreasing algorithm (MBFD) to map VMs into Physical Machines (PMs), seeking to reduce power consumption. The Best Fit Decreasing (BFD) algorithm inserts *items*, in *bins*, sorted in decreasing order of size. After, chooses the bin that will provide the minimum empty space after the item is inserted.

On the other hand, Xu et al. in [18], involves three strategies to *Extract* VMs: (i) find the VM with the greatest CPU Load and less memory, to reduce migration overhead (ii) identify the VMs that are competing for scarce resources and (iii) check the possibility to remove every VM and shut down the PM. Then, the extracted VM is migrated to the PM which provides the greatest utility increase. Despite the fact that they consider different strategies during the extraction phase, their method is still bounded and, consequently, limited to those specific objectives.

Contrariwise, to avoid theses dependences and limitations, our proposal aims to be flexible enough to enable the usage of any kind of objective, strategy or affinity, by representing them as Qualifiers. Then, our Framework uses those Qualifiers to Provide placements to new VMs and Organize the Cloud.

Likewise, other works also propose frameworks to generalise and/or facilitate the studies in this area. However, many were still bounded to specific goals, e.g., [12]. Next, we discuss some proposals, which are the exceptions.

The *Snooze* framework [19] proposes a dynamic hierarchical self-organising structure. It deploys agents in PMs, which have a partial view of the system, in order to make decisions about their own hosted VMs, such as migrate, resize, start, suspend, resume, shut down and destroy them. In this framework, objectives, triggers and scheduling are represented as centralised open defined policies, submitted to the PM agents, where they react according to the triggered events. The fact of divide and distribute the decisions across the PMs, induce the improvement of local scenarios, hoping to improve the whole Cloud. Therefore, it lacks the improvement of the Cloud as a single unit, i.e., lacks a holistic view.

The Cielo Framework [20], aims at aiding Cloud operators to adapt the resource allocation to applications to the operational conditions in a Cloud. It adopts a limited set of strategies to manage CPU allocation, Response Time, Power consumption, workload distribution and Bandwidth allocation. Then, it uses an *Evolutionary Game Strategy* to improve the Cloud. Basically, it treats applications as Players, which can play pre-defined strategies in their turn, to improve specific aspects of themselves. However, it adopts placement changes regardless of its effects across the other elements.

 ISBN: 978-1-61208-450-3

III. CLOUD MODEL

In this section, we describe a Cloud model for VMP problems and its elements. These elements are the key components of the proposed framework.

In Clouds, a *Placement* is defined as a possible relation between a VM and a PM, guest and host respectively. A *Scenario*, in its turn, is a set of placements, which defines where each VM is placed on the Cloud, representing the real state of the Cloud or a possible one.

As presented in Section I, real Cloud environments have Multiple Objectives. In our model, MOs can be represented by the following elements: Rules, Qualifiers, Priorities and Cost.

Rules define placement constraints, i.e., rules can forbid VMs to be placed in PMs. They are divided into two types: Rules Free of Context (RFC) and Rules Sensitive to the Context (RSC). The context refers to a given scenario, which is the context where the VMs are placed. RFC define whether a given VM can be hosted by a given PM and RSC define whether a given scenario is valid or not, i.e., RFC validates placements and RSC validates scenarios. In order to illustrate these rules, consider the following examples: (i – RSC) a VM A should not be hosted in the same PM as a VM B, and (ii – RFC) a VM A must be placed in a PM with a specific architecture of processor.

Qualifiers are functions used to assess the quality of one or more placements of a Scenario. These assessments will enable the comparison and, consequently, selection of scenarios.

Priorities are applied to sort the qualifiers. They define weights and preferences between them, avoiding conflicts.

Cost is a function that quantifies the implementation cost between two scenarios, a root and a target scenario. Its values disregard units and can vary from the number of migrations to the required resource to implement the target scenario. Differently from the Qualifiers, it is not limited to a range of values and it depends of two scenarios to process a result.

This model was inspired by the model used in [21], which is based on the Organisation Theory.

TABLE I. SYMBOLS USED IN THE CLOUD MODEL DESCRIPTION

Symbol	*Meaning*
y	Number of VMs
x	Number of PMs
v	VM id
h	PM id
V	Set of y VMs
H	Set of x PMs
(V_a, H_b)	A placement
C	Set of y VM Placements
rf	Rule Free of Context (RFC)
rs	Rule Sensible to Context (RSC)
Rf	Set of RFC
Rs	Set of RSC
q	Qualifier Function
z	Number of Qualifiers
Q	Set of z Qualifiers
U	Qualifier's weights
i	Cost Function
m	Maximum Cost

A. Formal Model

To avoid ambiguity and to precisely specify the elements involved in the MO driven Clouds, in this section we formally define the model previously described. Table I presents the symbols used in this section and their meanings.

Let y be the number of VMs and x be the number of PMs in the Cloud. V is a set with y VMs v and H is a set with x PMs h, as shown in (1). A Placement is a relation between any two elements from V and H, such as (v_a, h_b), and a configuration C is a set with y placements, $\{(v_a, h_b)_1, ..., (v_b, h_c)_y\}$, representing a scenario.

In the formal model, RFC and RSC are represented by two distinct sets of functions, Rf and Rs, which contains an amount of f and s boolean functions each, respectively. Given a Placement (v_a, h_b), a function rf returns 1 for permitted or 0 otherwise , as shown in (2). Given a Configuration C, a function rs returns 1 for permitted or 0 otherwise, as in (3).

Qualifiers, instead, are presented by Q, which is a set with z functions q, $Q = \{q_1, ..., q_z\}$. Each function q, given any Configuration C, returns a vector with y qualifications i, one for each placement in C, as given by (4). These qualifications are mapped between zero and two, excluding zero, i.e., $]0, 2]$. Zero is not included because a qualifier with this value is a RSC, which forbids a scenario.

To sort the qualifiers, there is a vector U referring to a set of weights, which are rational numbers equal or grater than zero, that prioritise the Qualifiers, as in (5).

Finally, i refers to Cloud's Cost Function, which given any scenario (C), returns a rational value grater than zero, related to the implementation Cost of the scenario C, as given by (6). Still, exists a number m that defines the maximum cost that the Managers are willing to spend.

$$V = \{v_1, ..., v_y\} \quad H = \{h_1, ..., h_x\} \tag{1}$$

$$Rf = \{rf_1, ..., rf_f\} \quad rf((v_a, h_b), \{0 \vee 1\}) \tag{2}$$

$$Rs = \{rs_1, ..., rs_s\} \quad rs(C, \{0 \vee 1\}) \tag{3}$$

$$q(C, \{p_1, ..., p_y\}) \wedge \forall p \in]0, 2] \tag{4}$$

$$U = \{u_1, ..., u_z\} \wedge \forall u \in (\mathbb{Q} \geq 0) \tag{5}$$

$$i(C, \{p_1, ..., p_y\}) \wedge \forall i \in (0 > \mathbb{Q} > m) \tag{6}$$

IV. Framework Design

This section describes the solutions proposed for VMP problems. Initially, we describe the steps of the organisation method, presenting also a use case. Then, we explain the provisioning method, which also uses the previous method. Finally, we discuss some aspects and peculiarities of our approach.

Considering the presented Cloud model, VMP methods must: (i) *maximises* the qualifier's evaluations and (ii) *minimise* the implementation cost. Our solution uses a recursive process, which receives the current state of the Cloud (current scenario) and seeks better scenarios by making migrations that increase its evaluations.

A. The Organising Method

This section explains in details each step of the developed methodology, while Figure 2 illustrates it.

Initiating: The method receives: (i) the current scenario, i.e., a vector with the real placements of the Cloud, (ii) an empty VM set to control recursions (*ignoreVms*) and (iii) a boolean indicator to sign recursions (*isMainRecursion*), starting as *true*. At the beginning, if all VMs were explored, it ends the recursion selecting the current scenario.

Placements Evaluation: The first step evaluates the received scenario and generate a VM's rank according to its assessments. Each Qualifier evaluates all placements, building an evaluation matrix $ES_{z,y}$. Then, each evaluation is raised to the power of its respective qualifier's weight. Afterwards, the set is reduced to a vector E_y by multiplying all evaluations of each VM. The reduction is represented by (7).

$$E_y \leftarrow \prod_{n=1}^{z} ES_{n,y}^{U_n} \tag{7}$$

VM Rank Selection: This step is based on the *Extraction* strategy and its heuristic selects the VM which: (i) has the lowest evaluation and (ii) is not present in the $ignoreVms$ set.

Generation of Non-Dominated Scenarios: The objective of this step is to identify where the selected VM could be migrated, and generate a scenario for each possibility, i.e., at most $x - 1$ scenarios. During this step, the following filters are applied: (i) the Rules (RFCs and RSCs, respectively), (ii) Max Cost filter, checking if the Scenario's cost is below the maximum, and (iii) a Pareto filter, which excludes dominated scenarios.

Search for new Scenarios: This step, based on the *Neighbourhood Search*, starts a new recursion to explore each Non-Dominated scenario found. However, to avoid loops the following information is sent with it: (i) a boolean signing *false*, meaning that is not the main recursion and (ii) a copy of the *ignoreVMs* set (*newIgnoreVMs*) and added the lowest VM, which was selected on the first step. The $newIgnoreVMs$ is used to avoid deadlocks during the Search.

Select Best Scenario: The goals of this step is to select the scenario which offers the best Cost-Benefit, also considering the current scenario among the recursion's results.

To achieve the equilibrium between the implementation cost and the qualifier's evaluation, we propose the Cost-Benefit method represented by (8). The benefit is defined as the difference between the evaluations of the current scenario of the Cloud (Real Scenario – RS) and a target scenario of the Cloud (Possible Scenarios – PS). More specifically, is the sum of the variations of each VM evaluation of RS and PS. Since PS is a non-dominant solution, it is not possible to have a negative variation. Then, the cost-benefit (cb) is calculated by the division of the benefit with the result of the cost function.

$$cb = \frac{\sum_{n=1}^{y} PS_n - RS_n}{i(PS)} \tag{8}$$

Return Decision: Finally, this step decides either to continue the search or return the selected scenario. If it is not the main recursion this step will return the best scenario, which was selected. Otherwise, it will *vary* the *Neighbourhood* by starting a new recursion with a non-empty set of VMs to be ignored. In this case, it will add the lowest placement of the selected scenario in the $ignoreVms$ set and starts a new recursion using this scenario and this VM set. The result of this recursion is the method's output. Lines 14-19 of Alg. **??** show this step.

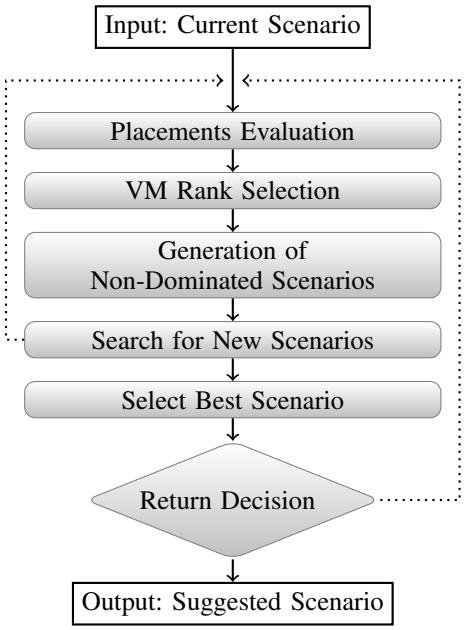


Figure 2. Flow Chart of the Organising Method

B. Provisioning for New VMs

The objective of this feature is to choose the best placement for a new VM. In this case, the best placement would be the one which provide the greatest benefit. This feature takes advantage of the Organising method to select the placement.

The first step is, based on the current scenario, to adopt a theoretical placement in the first allowed PM. Then, from a set with all VMs of the Cloud, the newcomer is extracted from it, in order to generate the set of VMs to be ignored. Before start the recursion, the Cost function is deactivated, meaning that there is no cost involved in the instantiation. After, an Organising recursion is started sending the theoretical scenario and the built set, as initial variables. The result of the recursion contains the placement of the new VM.

 ISBN: 978-1-61208-450-3

C. Discussion

In this section we discuss some aspects and differences regarding other proposals and implementation issues.

The Cloud initial consistency, with the environment Rules, is demanded for the proper work of the method, which means that the current state of the Cloud must be in accordance with the RFCs and RSCs. The adoption of an initial illegal scenario interferes with the generation of new scenarios. For this, a pre reorganisation process must be applied, which is considered out of the scope of this work.

Adaptations of Variable Neighbourhood Search (VNS) and Pareto's Dominance were adopted in this proposal. The VNS strategy builds a Pareto Front by recursively searching Non-Dominated results from different neighbourhoods. However, instead of using random scenarios to vary the searched Neighbourhood, as proposed in [10], we use a method based on the proposed VM Rank to vary the starting point of the search.

Likewise, the original definition of Pareto's Dominance compares the objectives assessments separately. We, instead, compare the placements' evaluations to avoid the method's stagnation due to quantity of objectives. Our judgement is based on the scenario's evaluations, which is a combination of all its qualifiers assessments and its weights.

The ignore VMs set is used mainly to avoid deadlocks. A deadlock occurs when the lowest VM is always the same, i.e., their evaluation growth depends on the relocation of other VMs. Thus, the next recursion will ignore this VM and will try to explore other placements.

Although its use together with the *Dominance* Search ensures the convergence of the method, it does not assure finding the general best scenario. For instance, cases where the temporary adoption of a degraded (dominated) scenario is needed to achieve the optimal one. However, it always leads to a better scenario, even if the method is forced to stop before reaching the end of the search.

Our current efforts are to focus on identify and test new deadlock scenarios, testing new approaches in that specific subject, such as: (i) using two independent $ignoreVMs$ sets; (ii) changing from Depth to Wide Search, pushing the Pareto Front forward; and (iii) using a Relaxation Factor strategy in the Pareto filter, enabling the acceptance of dominated results with great benefits.

Qualifiers based on Monitoring data, in some cases, are demanded to evaluate placements. However, Evolutionary algorithms compute many scenarios per second, which makes it infeasible to depend on real-time data. For that reason, it is recommended to pre-load the needed data and/or utilise approximation functions in the Qualifiers, if needed.

Provisioning Gaps: Despite the proposed strategy works in practically all cases, it's based on the fact that there are resources available during the first phase. We acknowledge this weakness, however, we consider very unlikely to happen, once a certain resources waste are maintained in the PMs.

The Max Cost constraint is one of the main convergence catalyser of the method, i.e. it bounds many unaffordable branches during the exploration, considerably decreasing the number of scenarios and accelerating the search. However, the use of this constraint should be based on the premiss that the cost does not decrease from a base scenario to its successors. Otherwise, valid branches could be discarded from the search.

V. Experiments

In this section, we present the experiments to assess some aspects of our proposal. To this aim, we use real placement data from the main data centre of the Federal University of Santa Catarina. This environment is composed of: 607 VMs, 18 PMs, 99 storage pools and 102 networks.

The experiments were divided into three categories: (i) *Performance*, (ii) *Selection* and (iii) *Filters*.

In *Performance*, we measure the necessary time to evaluate the real scenario and propose better placements. We vary the number of VMs (between 350 and 600) and the maximum number of migrations allowed (5, 10 and 20).

In *Selection*, to understand the advantages of the Rank Selection approach, which improves the worst placements of the scenario, we compare it to: (i) a method based on [20], called Turn Selection, which shuffles the VMs and tries to improve the VMs in turns, and (ii) a method called Highest Selection, which selects VMs on top of the Rank to relocation.

In *Filters*, to show the importance of the Dominance filter, i.e., a method that accepts only scenarios in which not a single placement's evaluation worsen and at least one improves in comparison to the previous scenario, we confront it to a filter that accepts scenarios which evaluations' sum are greater than the previous scenario, named Greater Benefit. The tests were executed in a scenario with 600 VMs and the migration's threshold were varied from 6 to 14.

For the experiments we implemented and employed the following rules: (i) pre-requirements for Live Migration, such as network and storage accessibility; (ii) resource availability for the migration; (iii) the cluster coherence for the migration, i.e., if the VMs are in the same PMs' cluster where they belong; and (iv) the implementation cost cannot exceed the limit. For the qualifiers, the following strategies were implemented: (i) Consolidate VMs in few PMs; (ii) Distribute nodes from the same services in different PMs; (iii) Distribute VMs of the same storage pool. For the Cost evaluation, we employed a function that regards the number of migrations necessary to achieve a given scenario.

The prototype was implemented in PHP 5.5 and the tests use the Test's Framework PHPUnit 4.2. The tests were executed in an environment with Ubuntu 14.04, an *Intel T9400* processor of 2.53GHz with 4GB of RAM. Which, in average, processed 200 scenario per second.

A. Implementation and Results Discussion

Performance: The worst case scenario, considering 600 VMs and 20 migrations, took 62 seconds to finish. The variation of the migration's threshold presented a greater impact when compared with the scenario's size. It considerably increases the Cost-Benefit Rate (+104%), number of analysed scenarios and, consequently, on the execution time (+7355%). Still, a linear pattern on execution time was noticed, due to this threshold, which forced the premature stop of the method, as shown in Figure 3(a).

Selection: On average, the Rank method, in comparison to the other methods, shown a greater Cost-Benefit raise on the lowest VMs, as shown in Figure 3(b). It was, on average, 72% faster than the TopRank method and increased 36% of the Cost-Benefit in the worst case scenario. Although this

 ISBN: 978-1-61208-450-3

behaviour was expected, since the Random and TopRank methods do not focus on the lowest VMs, the Rank method still managed to raise 2% of the average Cost Benefit of the worst case scenario, which was not expected.

Filters: As expected, when we increased the maximum amount of migrations the Dominance Filter prevented the exponential growth of explorations, still maintaining a similar Average Cost-Benefit (+0,002%) in comparison to the other approach. On the other hand, the Greater Benefit Filter raised exponentially the number of searches turning the execution time impractical, as shown in Figure 3(c). Exponential Regression leads us to an approximation of $y = 0.1832e^{0.47x}$, where x is the number of migrations and y the seconds to execute. Otherwise, Pareto's method give us a growth of $y = 1.76e^{0.196x}$, which is an acceptable for real environments.

In summary, due to the Dominance Filter the scenario's growth was drastically reduced and the Rank approach ensured that the gained benefits were focused on the main issues, the lowest placements. Finally, the performance results showed that overall our method is applicable in real Cloud environments, providing a better scenario in an acceptable time.

VI. Final Considerations and Future Works

In this paper, we propose a novel and flexible framework focus on organisation for VMP. To the best of our knowledge, this is the first work to combine multiple objectives evaluations with implementation costs to organise the VM in the Cloud.

The proposed framework supports different types of objectives, SLAs, strategies, best practices and costs in the form of qualifier, rule and cost functions and priorities. In order to demonstrate the advantages of our solution, we conducted several experiments, comparing it also with other approaches, and showed that our solution proposes a better scenario in a reasonable time.

In the future, we plan to: (i) implement a module to adapt Cloud's scenarios out of accordance with the environment rules; (ii) extend the *Cost-Benefit* method to consider multiple Costs during the selection phase; and (iii) implement rules and qualifiers that retrieve their logics from formal languages, such as SLAC [22].

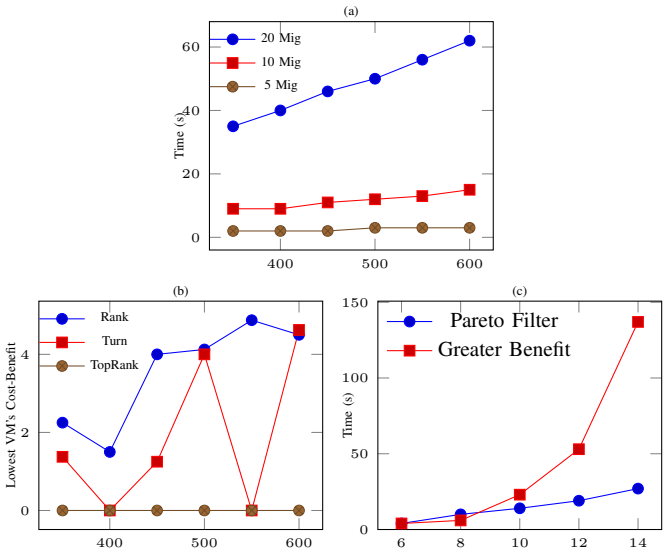


Figure 3. (a) Migrations Thresholds, (b) Selection Methods in Different Scenarios and (c) Filter Methods with Different Migrations Thresholds.

References

[1] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in Symposium on Cloud Computing – SoCC. ACM, 2010.

[2] S. Nathan, P. Kulkarni, and U. Bellur, "Resource availability based performance benchmarking of virtual machine migrations," in I.C. on Performance Engineering – SPEC. ACM, 2013.

[3] S. Abar, P. Lemarinier, G. K. Theodoropoulos, and G. M. OHare, "Automated dynamic resource provisioning and monitoring in virtualized large-scale datacenter," in I.C. on Advanced Information Networking and Applications – AINA. IEEE, 2014.

[4] A. Beloglazov and R. Buyya, "Energy efficient allocation of virtual machines in cloud data centers," in I.C. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2010.

[5] C. C. T. Mark, D. Niyato, and T. Chen-Khong, "Evolutionary optimal virtual machine placement and demand forecaster for cloud computing," in I.C. on Adv. Info. Net. and Applications – AINA. IEEE, 2011.

[6] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in Green Computing and Communications – GreenCom / CPSCom. IEEE, 2010.

[7] L. Wang, J. Xu, M. Zhao, and J. Fortes, "Adaptive virtual resource management with fuzzy model predictive control," in I.C. on Autonomic Computing – ICAC. ACM, 2011.

[8] O. Abdul-Rahman, M. Munetomo, and K. Akama, "Toward a genetic algorithm based flexible approach for the management of virtualized application environments in cloud platforms," in I.C. on Computer Communications and Networks – ICCCN. IEEE, 2012.

[9] C. von Lucken, B. Barán, and C. Brizuela, "A survey on multi-objective evolutionary algorithms for many-objective problems," in Computational Optimization and Applications. Springer, 2014.

[10] Y.-C. Liang and M.-H. Lo, "Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm," Journal of Heuristics, 2010.

[11] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," Transactions on Evolutionary Computation, 2003.

[12] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," Computer Networks, 2013.

[13] P. Calyam, R. Patali, A. Berryman, A. M. Lai, and R. Ramnath, "Utility-directed resource allocation in virtual desktop clouds," Computer networks, 2011.

[14] J. Chen, K. Chiew, D. Ye, L. Zhu, and W. Chen, "Aaga: Affinity-aware grouping for allocation of virtual machines," in I.C. on Advanced Information Networking and Applications – AINA. IEEE, 2013.

[15] O. Biran et al., "A stable network-aware vm placement for cloud systems," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2012.

[16] D. Dong and J. Herbert, "Energy efficient vm placement supported by data analytic service," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2013.

[17] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," Special Interest Group on Operating Systems – SIGOPS, 2001.

[18] J. Xu and J. Fortes, "A multi-objective approach to virtual machine management in datacenters," in I.C. on Autonomic Computing – ICAC. ACM, 2011.

[19] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in I.S. on Cluster, Cloud and Grid Computing – CCGrid. IEEE/ACM, 2012.

[20] Y. Ren, J. Suzuki, A. Vasilakos, S. Omura, and K. Oba, "Cielo: An evolutionary game theoretic framework for virtual machine placement in clouds," in I.C. on Future Internet of Things and Cloud – FiCloud. IEEE, 2014.

[21] G. A. Geronimo, J. Werner, C. B. Westphall, C. M. Westphall, and L. Defenti, "Provisioning and resource allocation for green clouds," in I.C. on Networks – ICN. IARIA, 2013.

[22] R. B. Uriarte, F. Tiezzi, and R. D. Nicola, "Slac: A formal service-level-agreement language for cloud computing," in I.C. on Utility and Cloud Computing – UCC. IEEE/ACM, 2014.

 ISBN: 978-1-61208-450-3

Influence of the Channel Model in the Optimum Switching Points in an Adaptive Modulation System

Ana Paula Teles Ribeiro da Silva and José Marcos Câmara Brito
Instituto Nacional de Telecomunicações - INATEL
Santa Rita do Sapucaí, Brazil
e-mail:anaptrs@gmail.com, brito@inatel.br

***Abstract*— Adaptive modulation techniques have been widely employed in order to improve the performance of wireless networks. The switching point between neighboring modulations is a key issue to define the performance of an adaptive modulation system. In this paper, we analyze in depth the influence of the fading channel model in the optimum switching points considering the maximum throughput as the criterion to determine the switching points.**

Keywords- Adaptive modulation; Nakagami-m fading; optimum switching points; maximum throughput criterion.

I. INTRODUCTION

Due to the growing demand for high transmission rates and the scarcity of spectrum in wireless networks, several studies have been developed to improve the performance and ensure Quality of Service (QoS) for such networks.

The adaptive modulation technique has been studied in order to improve the performance of time-varying conditions channels, maintaining the required performance [1]-[3]. This scheme consists of the dynamic adaptation of the modulation scheme as a function of the channel's state. The receptor makes an estimation of the channel state and sends this information back to the transmitter through a feedback channel. Based on this information, the transmitter changes the modulation order, choosing a modulation that better matches with the conditions of the channel at that time [1]-[5].

In general, the change of modulation order occurs between neighboring modulations (for a modulation with 2^n points in its constellation, the neighboring modulations has 2^{n-1} or 2^{n+1} points in its constellation). A key issue in adaptive modulation schemes is to define the best point to switch from one modulation to its neighboring modulation [4][5].

The switching points between neighboring modulations can be determined in several ways, and the most commonly used in the literature is to determine the switching as a function of a target for the bit error rate (BER) [2], or a target for the packet error rate (PER) in the channel [1]. However, in [4] the authors show that these criteria can't optimize some quality of service parameters, like throughput; thus, these authors propose the determination of the optimum switching points based on the maximum throughput criterion in the wireless channels. The analysis presented in [4] consider a memory-less channel, e.g., an AWGN channel. Then, in [5], the authors use the same criterion, but considering Rayleigh fading channels, and show that the model of the channel affects the optimum switching points.

In this paper, we analyze in depth the influence of the channel model in the optimum switching points of an adaptive modulation scheme. For this, we consider the more generic Nakagami-*m* channel model. We consider a wireless network over a Nakagami-*m* block fading channel with adaptive *M*-ary Quadrature Amplitude Modulation (*M*-QAM). Thus, this paper extends the analysis presented in [5] considering several different fading channels, besides the Rayleigh fading model considered in [5].

The remainder of this paper is organized as follows: In Section II, we introduce the channel model; in Section III, we present the calculation of the exact PER in the channel, necessary to compute the throughput; the maximum throughput criterion is shown in Section IV; Section V presents the numerical results, and finally we present our conclusions in Section VI.

II. CHANEL MODEL

We assumed that the transmissions occur over a slowly-varying flat-fading channel, modeled as a Nakagami-*m* block fading channel, where the choice of modulation in the physical layer is made on a frame-by-frame basis, so that the channel gain remains invariant on a single frame, but may vary between adjacent frames [1][3].

The signal-to-noise ratio (SNR) in the receiver can be statistically described by a general Nakagami-*m* model, with probability density function (pdf) given by:

$$p_\gamma(\gamma) = \frac{m^m \gamma^{m-1}}{\left(\bar{\gamma}\right)^m \Gamma(m)} \exp\left(-\frac{m\gamma}{\bar{\gamma}}\right). \quad (1)$$

where $\bar{\gamma}$ is the average received SNR, $\Gamma(m)$ is the Gamma function defined by $\Gamma(m) = \int_0^\infty x^{m-1} e^{-x} dx$, and m is the Nakagami fading parameter [1].

The Nakagami-*m* fading model is equivalent to a set of independent Rayleigh fading channels obtained by maximum ratio combining (MRC), with the diversity order of *m* [6].The Nakagami-*m* distribution is an approximation widely used in literature for representing a wide range of multipath

 ISBN: 978-1-61208-450-3

channels [2]. For $m = 0.5$, the Nakagami fading model represents the unilateral Gaussian distribution (which corresponds to the highest amount of multipath fading scenario). When $m = 1$, it assumes a Rayleigh distribution model. For $m > 1$, there is a one-to-one mapping between the Nakagami fading parameter and the Rician factor and allows the Nakagami distribution to approach the Rice distribution [2]. Furthermore, reference [2] claim that the Nakagami-*m* distribution often gives the best fit to urban and indoor multipath propagation.

Thus, in this paper, we study the impacts of the variation of the parameter *m* in the optimum switching points in an adaptive modulation scheme.

III. CALCULATION OF THE EXACT PER

A. Instantaneous PER

We employed several *n*-mode modulation schemes in our analysis: *M*-QAM with *M* = 8, 16, 32, 64, 128 and 256. Each *M*-ary modulation scheme has R_n bits per symbol, where and n = 1, 2… 6 the modulation mode (one of the six considered modulations).

The data is transmitted in packets containing a fixed number of bits, called n_p. Therefore, in each modulation mode each packet is mapped in n_p / R_n symbols [1].

For a system where the bits inside a packet have the same BER and the bit-errors are uncorrelated, the PER can be calculated as a function of the BER, as in [1]:

$$PER = 1-\left(1-BER\right)^{n_p}. \tag{2}$$

However, the author of [1] claims that the PER calculation through BER in (2) isn't accurate for large-size QAM constellations because the information bits of the same packet incurs different error probabilities for such constellations.

In this paper, we use the approach proposed in [1] to compute the PER in the system, which, in turn, is based on the methodology proposed in [7] to compute the exact BER for an arbitrary rectangular *M*-QAM modulation.

To compute the BER, following [7], we consider that a rectangular *M*-QAM modulation can be modeled as two independent pulse amplitude modulation (PAM), *I*-ary and *J*-ary, where $M = I * J$. The exact BER expression is obtained by observing regular patterns that occur due to the characteristics of Gray code bit mapping. The error probability of the *k*th bit in-phase components of the *I*-ary PAM, where $k \in \{1, 2, ..., \log_2 I\}$, is given by [7]:

$$P_I(k) = \frac{1}{I}\sum_{i=0}^{(1-2^{-k})I-1}\left\{(-1)^{\left\lfloor \frac{i\cdot 2^{k-1}}{I}\right\rfloor}\left[2^{k-1}-\left\lfloor \frac{i\cdot 2^{k-1}}{I}+\frac{1}{2}\right\rfloor\right]\times \operatorname{erfc}\left((2i+1)\sqrt{\frac{3\log_2(I\cdot J)\cdot\delta}{I^2+J^2-2}}\right)\right\} \tag{3}$$

where δ denotes the SNR per bit, and $\lfloor x \rfloor$ denotes the largest integer to *x*.

The error probability of the *l*th bit in the quadrature components of the *J*-ary PAM, where $l \in \{1, 2, ..., \log_2 J\}$, is obtained from [7]:

$$P_J(l) = \frac{1}{J}\sum_{j=0}^{(1-2^{-l})J-1}\left\{(-1)^{\left\lfloor \frac{j\cdot 2^{l-1}}{J}\right\rfloor}\left[2^{l-1}-\left\lfloor \frac{j\cdot 2^{l-1}}{J}+\frac{1}{2}\right\rfloor\right]\times \operatorname{erfc}\left((2j+1)\sqrt{\frac{3\log_2(I\cdot J)\cdot\delta}{I^2+J^2-2}}\right)\right\}. \tag{4}$$

In [1], the authors use the results given by (3) and (4) and derive an exact closed-form to compute PER. The exact instantaneous PER for each modulation mode as a function of the received SNR in a system with rectangular QAM symbols is given by [1]:

$$PER_n(\gamma) = 1-\left\{\prod_{k=1}^{\log_2 I}\left[1-P_I(k)\right]^{(n_p/\log_2(I\cdot J))}\times\prod_{l=1}^{\log_2 J}\left[1-P_J(l)\right]^{(n_p/\log_2(I\cdot J))}\right\}. \tag{5}$$

To compute $P_I(k)$ and $P_J(l)$ we considered $I = J = \sqrt{M}$ for square QAM modulations (256, 64 and 16-QAM), $I = 8$ and $J = 16$ for 128-QAM, $I = 4$ and $J = 8$ for 32-QAM, and finally $I = 2$ and $J = 4$ for 8-QAM.

B. Average PER

As stated earlier, in this system the packets are transmitted over a block-fading channel and, to determine the average PER, it is necessary to consider the influence of the fading. So, the average PER for each modulation mode is equal to the integral of the instantaneous PER for the current modulation *n*, multiplied by the probability density function of the received SNR (in this case, the pdf of a Nakagami-*m* distribution)[1][8][9]. Therefore, the average PER is:

$$\overline{PER_n(\gamma)} = \int_0^{\infty} PER_n(\gamma)\, p_\gamma(\gamma)\, d\gamma. \tag{6}$$

IV. THE MAXIMUM THROUGHPUT CRITERION

Since we considered an adaptive modulation scheme without coding, all transmitted bits will be information bits. We consider only correct packets to compute the throughput (some authors refer to this as goodput). Thus, to compute the normalized throughput of the current modulations, we consider the following factors [4]:

- The ratio between the maximum numbers of transmitted bits and the maximum number of possible bits. In other words, the number of bits per symbol of the current modulation over the number of bits per symbol of a reference modulation (in our case, 256-QAM);
- The percentage of packets correctly transmitted.

The normalized throughput of the current modulation is given by:

$$\eta = \frac{\log_2 M_i}{\log_2 M_r} \cdot P_c \qquad (7)$$

where M_i is the number of points in the constellation of the current modulation, M_r is the number of points in the constellation of the reference modulation and P_c is the probability of a packet being correctly transmitted, given by, $P_c = (1 - PER)$.

V. NUMERICAL RESULTS

In this section, we present numerical results for the optimum switching points using the maximum throughput criterion. We considered 256-QAM as the reference modulation and we set the packet length, following [5], n_p = 424 bits. We analyze the influence of the fading on the optimum switching points by varying the diversity order m of the fading Nakagami considering m = 0.5, 1, 2, 3 and 10 (this value was used in order to consider an AWGN-like channel).

To compute the throughput, for a given modulation and a given average SNR, we compute the PER using (3), (4), (5) and (6), then we compute the throughput using (7). All computations were done using Mathcad software.

Figures 1, 3, 5, 7 and 9 show the throughput curves, as a function of the average SNR, for m = 0.5, 1, 2, 3 and 10, and modulations 256, 128, 64-QAM, respectively. Figures 2, 4, 6, 8 and 10 show the throughput curves, as a function of the average SNR, for the same m = 0.5, 1, 2, 3 and 10, but now considering the modulations 64, 32, 16 and 8-QAM, respectively.

The optimum switching point between two neighboring modulations is obtained by the crossing point of the corresponding throughput curves.

Table I shows the optimal switching points and the throughput in these points. We can observe that the throughput in the switching points increases with the increasing of the diversity order m. So, in a system with adaptive modulation at the physical layer, if the diversity order increases, i.e., the channel becomes less severe in terms of fading, the throughput in the switching points improves.

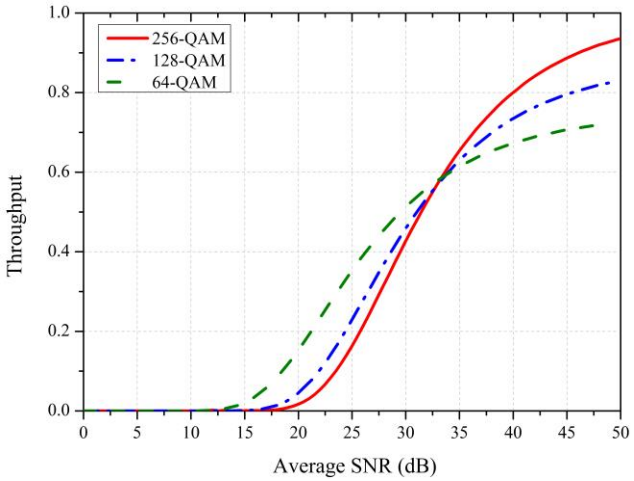


Figure 1. Throughput for m = 0.5 and 256, 128 and 64-QAM.

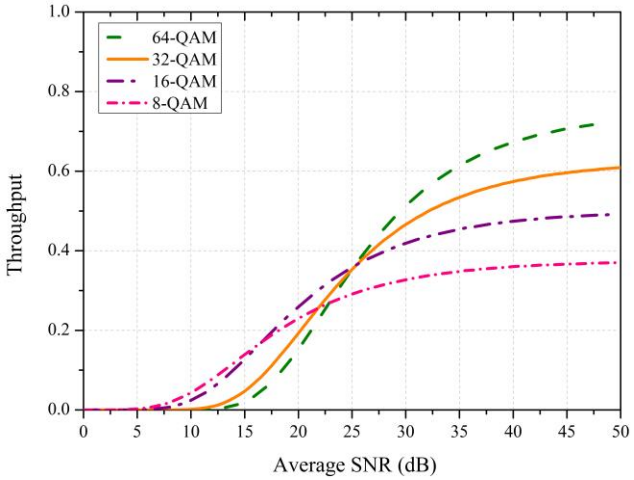


Figure 2. Throughput for m = 0.5 and 64, 32, 16 and 8-QAM.

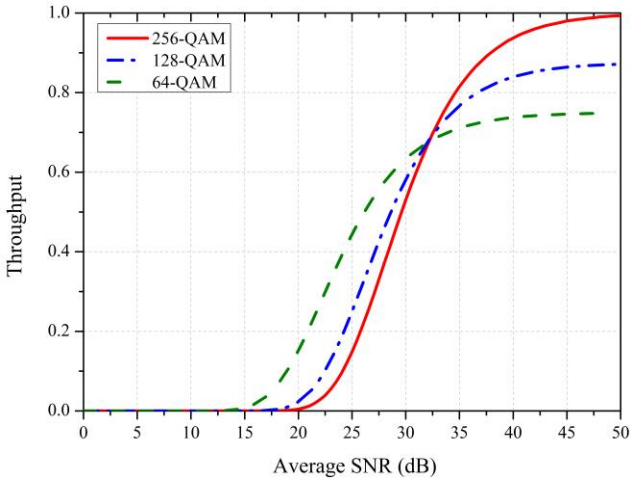


Figure 3. Throughput for m = 1 and 256, 128 and 64-QAM.

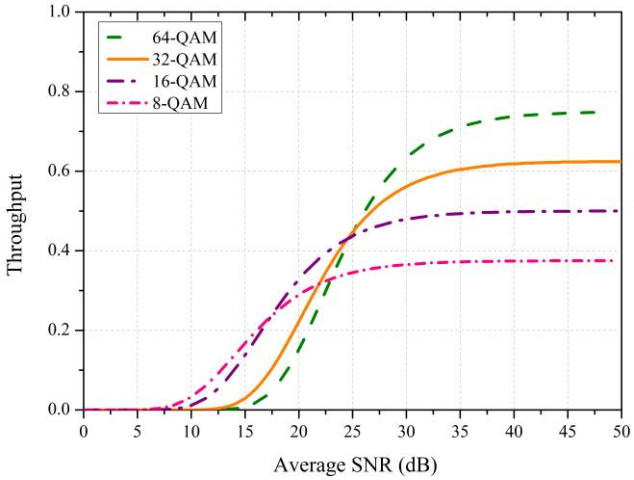


Figure 4. Throughput for m = 1 and 64, 32, 16 and 8-QAM.

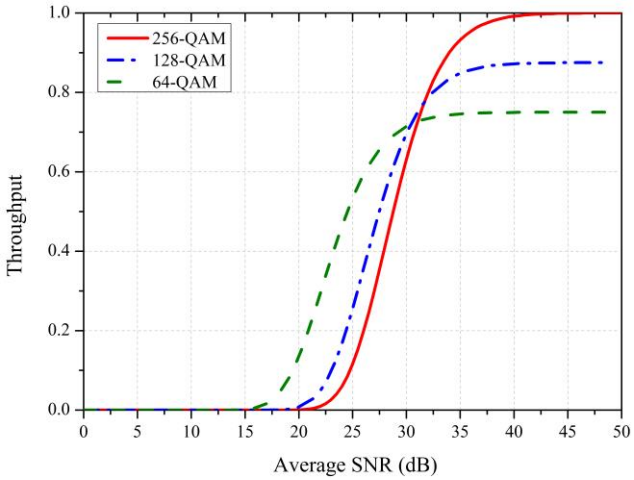


Figure 5. Throughput for m = 2 and 256, 128 and 64-QAM.

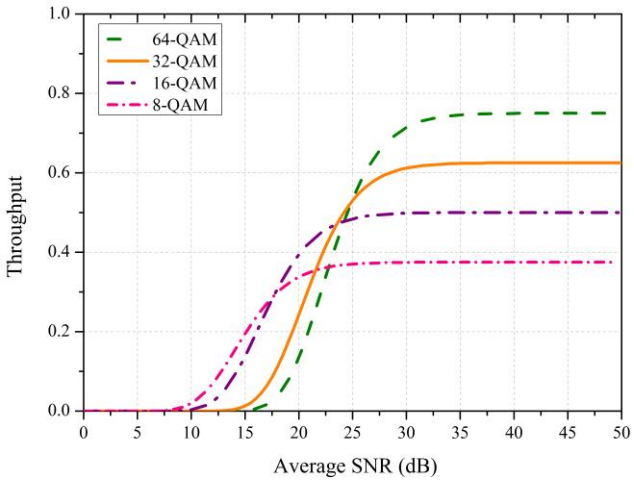


Figure 6. Throughput for m = 2 and 64, 32, 16 and 8-QAM.

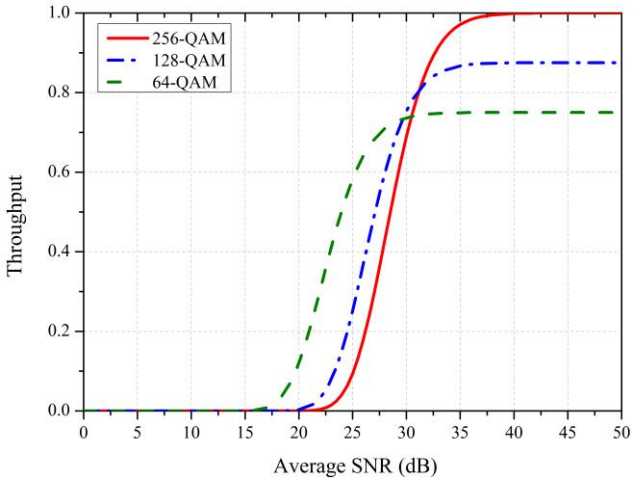


Figure 7. Throughput for m = 3 and 256, 128 and 64-QAM.

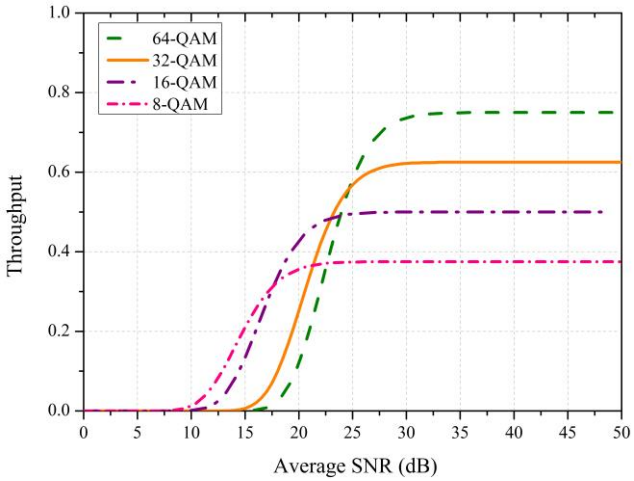


Figure 8. Throughput for m = 3 and 64, 32, 16 and 8-QAM.

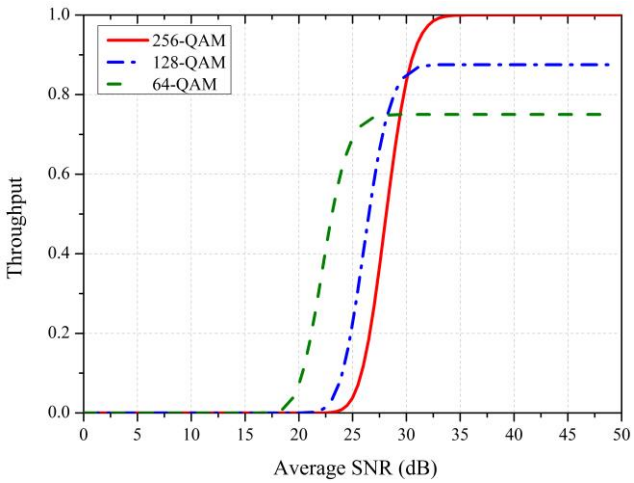


Figure 9. Throughput for m = 10 and 256, 128 and 64-QAM.

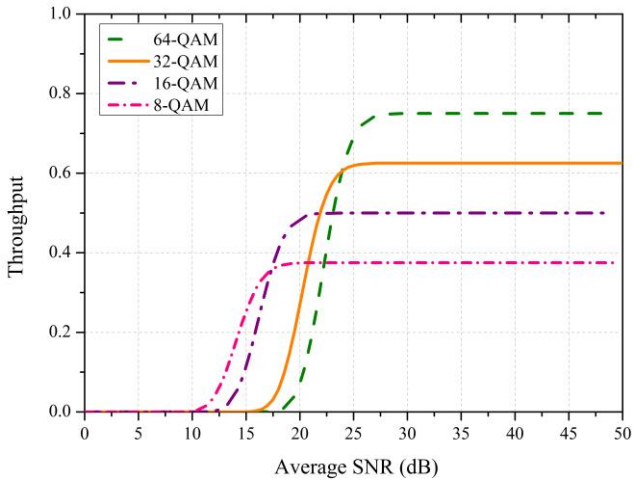


Figure 10. Throughput for m = 10 and 64, 32, 16 and 8-QAM.

TABLE I. OPTIMUM SWITCHING POINTS AND THROUGHPUT

Switch from	***m***	**Average SNR (dB) Switching points**	**Throughput (η)**
256 to128-QAM	0.5	32.7	0.559
	1	32.3	0.685
	2	31.6	0.77
	3	31.2	0.806
	10	30.1	0.846
128 to 64-QAM	0.5	33.7	0.592
	1	31.9	0.671
	2	30.4	0.718
	3	29.6	0.729
	10	28.2	0.746
64 to 32-QAM	0.5	25.1	0.353
	1	25.1	0.449
	2	24.9	0.527
	3	24.6	0.553
	10	23.9	0.603
32 to 16-QAM	0.5	25.4	0.364
	1	24.5	0.429
	2	23.5	0.469
	3	23	0.484
	10	21.8	0.492
16 to 8-QAM	0.5	16.5	0.168
	1	17.3	0.231
	2	17.6	0.289
	3	17.6	0.311
	10	17.3	0.353

Also, we can see that the optimum switching points vary with the channel model (represented by the parameter m of the Nakagami model). Also, for a particular value of m, we can observe that the SNR at the switching points is not a fixed value, but varies with the neighboring modulations.

Finally, we can observe that the switching points between some neighboring modulations are close to each other. For example, the switching point between the neighboring modulations 256-QAM to 128-QAM is close to the switching point between the neighboring modulations 128-QAM to 64-QAM, for m = 1, 2, 3 and 10. The switching point between the neighboring modulations 64-QAM to 32-QAM is also close to the switching point between the neighboring modulations 32-QAM to 16-QAM, and, therefore, their throughput is close as well, for all considered m. Based on this result, we can conclude that some modulations (like 128-QAM and 32-QAM) can't be considered in the implementation of an adaptive modulation system.

Furthermore, for $m = 0.5$ the switching point from 128 to 64-QAM occurs before the switching point from 256 to 128-QAM, similarly, the switching point from 32 to 16-QAM occurs before the switching point from 64 to 32-QAM. Thus, 128-QAM and 32-QAM should not be used in the adaptive modulation system in this case.

VI. CONCLUSION

In this paper, we considered a system with an adaptive modulation technique, considering a Nakagami-m fading channel model, and we analyzed the optimum switching points between neighboring modulations by using the maximum throughput criterion. We considered M-QAM modulations, with M = 8, 16, 32, 64, 128 and 256, and we set the diversity order to m = 0.5, 1, 2, 3 and 10.

We observed that if we modify the diversity order m of the Nakagami fading model, the optimum switching points, and the throughput in the switching points changed. So, the higher is the parameter m, higher is the throughput.

We also observed that some switching points between neighboring modulations are close to each other, indicating that some modulations (like 128-QAM and 32-QAM) can be neglected in the implementation of a practical adaptive modulation scheme.

ACKNOWLEDGMENT

This work was partially supported by Finep, with resources from Funttel, Grant No. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações - CRR) project of the National Institute of Telecommunications (Instituto Nacional de Telecomunicações - Inatel), Brazil.

REFERENCES

[1] Q. Liu, S. Zhou, and G. B. Giannakis, "Cross-Layer combining of adaptive Modulation and coding with truncated ARQ over wireless links," IEEE Trans. Wirel. Commun., vol. 3, no. 5, Sep. 2004, pp. 1746–1755.

[2] M.-S. Alouini and A. J. Goldsmith, "Adaptive Modulation over Nakagami Fading Channels," Wirel. Pers. Commun., vol. 13, no. 1–2, May 2000, pp. 119–143.

[3] T. Quazi and H. Xu, "Performance analysis of adaptive M-QAM over a flat-fading Nakagami-m channel," South Afr. J. Sci., vol. 107, no. 1–2, Feb. 2011, pp. 1–7.

[4] J. M. C. Brito and I. S. Bonatti, "Analysing the optimal threshold level for adaptive modulation in the wireless ATM networks," Proc Iasted Int. Conf. Wirel. Opt. Commun. Banf Can., Jul. 2002, pp. 510–515.

 ISBN: 978-1-61208-450-3

[5] M. S. Y. Bandiri and J. M. C. Brito, “Analyzing the Optimum Switching Points for Adaptive Modulation in Wireless Networks with Rayleigh Fading,” 6th IEEE Lat.-Am. Conf. Commun, Nov. 2014, six pages.

[6] Z. Wang and G. B. Giannakis, “A simple and general parameterization quantifying performance in fading channels,” IEEE Trans. Commun., vol. 51, no. 8, Aug. 2003, pp. 1389–1398.

[7] K. Cho and D. Yoon, “On the general BER expression of one- and two-dimensional amplitude modulations,” IEEE Trans. Commun., vol. 50, no. 7, Jul. 2002, pp. 1074–1080.

[8] Y. Xi, A. Burr, J. Wei, and D. Grace, “A General Upper Bound to Evaluate Packet Error Rate over Quasi-Static Fading Channels,” IEEE Trans. Wirel. Commun., vol. 10, no. 5, May 2011, pp. 1373–1377.

[9] A. J. de Faria and J. M. C. Brito, “A New Throughput Analysis in Cognitive Radio Networks Using Slotted CSMA,” presented at the COCORA 2013, The Third International Conference on Advances in Cognitive Radio, Apr. 2013, pp. 1–6.

 ISBN: 978-1-61208-450-3

Towards a Method Integrating Virtual Switch Performance Into Data Centre Design

Mitalee Sarker* and ‡, Jan Siersch†, Arslan Khan* and Stefan Wesner* and †
*Institute of Information Resource Management
Ulm University, Germany
Email: [firstname.lastname]@uni-ulm.de
†Kommunikations und Informationszentrum (kiz)
Ulm University, Germany
Email: stefan.wesner@uni-ulm.de
‡ Landeshochschulnetz BelWü, Germany
Email: mitalee.sarker@belwue.de

***Abstract*—Data Centre Design is a complex task due to the wide range of technological options and building blocks available. Deriving the data centre system architecture is done based on a couple of uncertain assumptions rather than known facts. The *future* workload of users or, more precisely, of their applications can only be predicted. Furthermore, the benchmarking and design process is typically driven by hardware features and options. In this paper, we argue that an important aspect to be considered in the design process is not only the network capabilities and topology but also the intended virtual switch solutions and their performance potentially jeopardising the overall system quality. The paper contains preliminary work based on a new network-aware algorithm for virtual machine placement in a virtualised cloud environment. Our initial evaluations help to create a fast and effective decision-making model for such an algorithm.**

Keywords–Data Centre Design; Virtual Switch; Software Defined Networking Performance.

I. INTRODUCTION

The design of a data centre is inherently complex and covers a wide range of technical disciplines. The task starts with considerations on the server capabilities, their specific configurations in terms of *Central Processing Unit* (CPU) types, frequency, number of cores, as well as all the other hardware components such as main memory, I/O system, network connectivity and network topology. At the end of the process, the facility infrastructure planning covers power supply and distribution, cooling approaches and optimisation, as well as specific floor plans describing the layout of the racks in rows within the computing room, and preparing the set-up and operation of the system. The latter part has gained particular attention in the last years as values for achieving power efficiency e.g., expressed with values, such as the Power Use Efficiency (PUE), have become an essential part of the design process.

Some data centre systems are designed for highly resource demanding applications such as for technical simulations or big data analytics. Before a procurement or buying decision for those data centres is done, it is common to realise a tailored data centre design. This design aims to deliver a system with a balanced server architecture by avoiding a system where one or a combination of several components limits the overall system performance. For example, as discussed in [1], this process requires an in-depth analysis of collected usage and performance data of a previous or similar systems, and varies significantly for different user groups and disciplines, even within a single application domain. Therefore, it is common to have no homogeneous system architecture, but a combined and jointly operated system with several segments that might differ significantly in their server architecture [2].

Many Cloud computing data centres are designed to cover a wide range of applications. They are optimised for low costs that can be best achieved with a system architecture that is based on simple components and is as homogeneous as possible. In this paper though, we focus on *heterogeneous* Cloud data centres that have optimised servers for hosting different types of *Virtual Machines* (VMs), and perform an active management not only for the initial placement of VMs but also for their migration to different servers based on observed performance, as proposed in [3]. This is particularly true if the targeted applications are resource demanding, such as *High Performance Computing* (HPC) simulations or High Performance Data Analytics (HPDA) [4] workloads. In traditional HPC data centres, the analysis and system design can be focused on the application workload and the operating system [5]. For a Cloud environment, the overhead of the hypervisors and, as we argue here, the network infrastructure play a key role and must be considered jointly with the hardware options. In other words, selecting the hardware system independently from the targeted virtualisation infrastructure can lead to severe bottlenecks and underutilisation.

We claim that, due to the currently observed performance characteristics of virtual switches, which are included in cloud environments such as OpenStack, have a severe impact on the overall data centre design. In particular, for Cloud data centres with demanding applications, the analysis of limiting capabilities of the underlying physical infrastructure is a common approach for selecting the most appropriate hardware for a given set of applications. This is commonly done by defining a set of benchmark applications that represents a typical workload in all of its aspects, from CPU load, over memory and external storage access patterns down to its communication behaviour. In general, these application-driven benchmarks are complemented by synthetic benchmarks. In order to provide a method to deliver comparable results across different system configurations that are considered as technological basis for a new data centre or, to find the most appropriate hosts within a given data centre, these synthetic benchmarks represent isolated elements with well understood and repeatable behaviour.

In a virtualised environment, in principle, the same approach can be followed with the extension that the analysis of

 ISBN: 978-1-61208-450-3

the physical hardware is no longer enough, but a set of virtual machines operating at the same time on a host in a defined mix has to be considered. Due to well advanced capabilities in hypervisors to virtualise CPU and memory resources, the loss of performance compared to a non-virtualised operation has decreased visibly with newer hardware generations, and has provided the foundation to include highly demanding applications within a virtualised environment.

The quality of the network connectivity plays a major role not only for the communication with other VMs, but also for accessing remote storage systems. Despite its key role in delivering overall performance, quite surprisingly, the network virtualisation is still often done with a purely software-based approach rather than relying on hardware level features.

In this paper, we introduce an initial model on how the performance limitations of virtual switch solutions can have an impact on the configuration options for a Cloud system architecture. In particular, these limitations represent a major caveat for demanding applications, and make the adoption of the Cloud model for such high performance applications less attractive.

A. Problem Statement

More specifically, the questions that need to be tackled in order to address the aforementioned issues are:

1) How could existing benchmarking and design approaches from HPC Data Centre design be extended or re-used to determine network bottlenecks in Cloud data centres?
2) Which methods can be used to identify performance properties of virtual switches and derive a model to predict the performance for different load situations?
3) How can such a model be used to validate the suitability of potential system architectures?

B. Related work

The work done in [6] is focused on VM placement and traffic routing in the data centre network in order to reduce the traffic cost. One online algorithm was introduced for the dynamic arrival and departure of VMs. Another online algorithm based on Markov approximation method was also presented by the authors, which performs a tradeoff between performance and cost. Although their work showed significant improvement on very large and small flows of data centre network, they have not considered the impact of the virtual switch on their proposed algorithms. In [7] Cohen et. al. discuss options for bandwidth-constrained VM placement optimisation problems. The algorithm is focused on storage area networks and is depicted from the communication between VMs and the core of the data centre network that connects the storage devices to the data centre network. Neither of these approaches has taken virtual switch constraints into account.

C. Solution Approach

The assumptions made for our approach are that the data centre considered is not a general purpose system, potentially hosting *all* kinds of applications with a completely *unpredictable* behavior, but certain knowledge is available about the types of applications hosted and their behavior. Based on these assumptions, application benchmarks creating a realistic and representative load can be derived, and synthetic benchmarks can be created by putting artificial load e.g., on the network to perform tests in a repeatable and consistent manner across different systems.

The remainder of this paper is structured as follows: Section II provides a theoretical model for the VM placement decision in a virtual switch aware Cloud data centre. Section III discusses our work on evaluating the performance characteristics of current virtual switch solutions for the network traffic model. Section IV presents the results gathered from these performance measurements, and interprets them. Lastly, Section V contains the conclusions drawn from this paper and an outlook on future work.

II. Network unaware VM placement

The challenge for an initial placement of VMs can be described as finding a distribution of k VMs $(v_1, v_2, \ldots, v_k)$ with a known maximum resource demand that is defined by a metric such as number of virtual cores, memory or disk space on a number of n servers with potentially different capabilities and load situation $S_1, S_2, \ldots, S_n$. Current models often focus on memory and compute core demand. If shared storage is used, no disk bottlenecks need to be considered with respect to capacity. A simplified algorithmic view assuming core and memory use as driving metrics is shown in the algorithm in Figure 1.

```
function SimpleCount(vmList)

    coresConsumed = 0
    memoryConsumed = 0
    vmCount = 0
5:  while vmList.current()≠ null do
        memoryConsumed += vmList[i].memoryReq
        coresConsumed += vmList[i].coreReq
        if currentMemory ≤ memMax
&& currentCoresUsed ≤ coreMax then
10:         vmCount +=1
            vmList.next()
        else
            // we exceeded the available resources...
            exit
15:     end if
    end while
    return vmCount
end function
```

Figure 1. Simple network agnostic VM packing algorithm

This almost naive algorithm is not uncommon in current Cloud implementations and sometimes only memory footprint of a VM is considered [8]. While it is definitely not a very good approximation of changing workload demands, it is commonly used as it maps nicely on the instance types. Instance types are the units that represent how resources within a cloud are typically offered. In addition, these metrics are well understood, as they relate to the typical selection criteria of physical servers in the past. Furthermore, for many applications in the data centre, the available network capacity of several 10Gbps, 40Gbps or even 100Gbps ports exceed the typical demand visibly. In order to address concerns related to security and potential influence

 ISBN: 978-1-61208-450-3

on performance on these shared network resources, virtual switches have been established as a common approach to act similarly to the hypervisor for CPU and memory resources as gatekeeper between different VMs.

We consider in this paper, in particular resource demanding applications in terms of communication. For example, due to significant I/O traffic on a network provided storage device, not only the performance of the network links but also the performance of the Virtual Switch becomes a concern.

A. Variety of virtual switch approaches

There are several virtual switch products currently available and they differ fundamentally in the way they are implemented. This section aims to provide a compact overview on the different switching approaches. In general, these switches consist of a data plane and a control plane. The data plane handles packet manipulation and contains the forwarding logic. The control plane manages the rules under which the switch operates through some sort of management protocol. In terms of implementation, there exists a spectrum from fully software-based switching approaches to integration of switching logic directly into hardware components. The detailed performance implications of these design decisions have yet to be evaluated.

Open vSwitch (OVS) [9] is an example of a purely software-based managed virtual switch [10], [11]. It is often used for network virtualisation in cloud environments such as OpenStack. Open vSwitch supports a variety of network virtualisation and management protocols. Fundamentally, the switch consists of a kernel space data plane and user space control plane. To avoid costly context switches between the kernel and user space components, flow caching is implemented as part of the kernel side logic of the switch.

The *Intel Data Plane Development Kit* (DPDK) [12] is a collection of user space application libraries dedicated to high-performance packet processing [13]. It improves performance by providing applications with multi-core enabled data plane functionality and poll mode Network Interface Card (NIC) drivers, which operate directly in user space. These drivers are available for 1GbE and 10GbE interfaces. *Intel DPDK vSwitch* is based on Open vSwitch, and modified to take advantage of the functionality provided by DPDK. This increases its packet switching performance, especially when handling large quantities of small packages [14], [15].

Lagopus [16] is a software-based, OpenFlow 1.3 compliant SDN switch with support for network function virtualisation and multiple data plane implementations [17]. Communication between the data plane and the switch agent takes place through an event queue mechanism. There exist multiple data plane implementations for the switch, such as raw sockets, Intel DPDK, and bare-metal switch variants that benefit from hardware acceleration.

In contrast to the previously mentioned products, the *Mellanox eSwitch* takes advantage of switching capabilities embedded in supported NICs [18], [19]. A user space daemon provides access to the switch's functionality. Moving the data plane directly into hardware yields the potential of improved performance compared to purely software-based solutions. In addition, it supports *single root I/O virtualisation* (SR-IOV) to multiplex multiple virtual interfaces (virtual functions) into a single physical interface (physical function). This, in combination with the embedded switching capabilities, removes the need for software-based virtual network devices for the connection of guest instances. This in turn leads to improved performance, while simultaneously reducing the CPU load of the physical host. In such a scenario, the eSwitch can handle packet flows between the physical and virtual functions, apply packet filtering rules (including protection against *Media Access Control* (MAC) spoofing), and isolate virtual networks using *Virtual Local Area Networks* (VLANs). However, in order to use eSwitch, specialised *Network Interface Cards* (NICs) are required, and the number of virtual functions available to the host is limited by the internal architecture of the NIC.

B. A virtual switch aware design method

As outlined in the previous section, different approaches exist to realise virtual switches. From purely software based over NIC aware up to NIC embedded switch solutions, one can also expect a wide range of different performance characteristics. As a result, the network demand, along with the current common metrics, needs to be included in the decision in order to make a suitable initial placement of VMs and perform migration and optimisation steps during operation.

Compute and memory demand can be considered as separate due to the capabilities of the underlying hypervisor. Although virtualisation can be achieved by using the virtual switches, the traffic produced by the different VMs is combined and is using the shared network resources per server. As a result, a more complex model for modelling the network demand is necessary where the overlay of the traffic from all VMs within a server needs to be considered.

Initially, consider the following model for a VM's network load as shown in Equation (1). It is very simple and it assumes that with a probability p, the VM_i is sending with a fixed rate of k kbps, and with probability $1-p$, no traffic is generated.

$$\mathrm{VM}_i = \begin{cases} k\ kbps & p \\ 0\ kbps & 1-p \end{cases} \tag{1}$$

A simple traffic model can be achieved from this initial model by using values for average bandwidth k and probability p for active and inactive periods, as well as an average time in state *active* and *inactive* based on traffic analysis. This traffic model can be used in a simulation to estimate average workload of different VMs combined at a virtual switch. It can be further extended by defining several states with different bandwidth demands and a matrix expressing the probabilities to move from a state i to a state j. With this model, we have assumed for the sake of simplicity, Markov properties that only the previous state is relevant. Furthermore, we can now define a matrix of probabilities Q for changing from state i to j as shown in Equation (2).

$$Q = \begin{pmatrix} 0 & q_{01} & \cdots & q_{0(n-1} \\ q_{10} & 0 & \cdots & q_{1(n-1)} \\ \vdots & \cdots & \ddots & \vdots \\ q_{(n-1)0} & \cdots & q_{(n-1)(n-2)} & 0 \end{pmatrix} \tag{2}$$

 ISBN: 978-1-61208-450-3

Together with the probability or time to stay in a state, we have a more fine grained model for the traffic that can be expected from a given VM_i. These probabilities can be either derived from the nature of the VM performing a white box modelling or, by observing the properties using traffic analysis tools and discretising the measured bandwidth to a defined number of different states and their changed probabilities.

In order to make a prediction, if a given server using a specific virtual switch is overloaded with a load of n VMs, one also needs to model the capabilities of the virtual switch as well. Again, a mix of the white box approach e.g., derive the overhead of certain methods to realise the tunnels, and black box modelling to determine how the outgoing bandwidth changes based on input load, has to be used. An initial model could be using the simple VM model with on and off states as a prediction, for how many VMs, a virtual switch can deliver the accumulated bandwidth as outgoing traffic. So the initial virtual switch model would correspond to the sum depicted in Equation (3).

$$\mathrm{BW}_{out} = \min\left\{\sum_{1}^{n} \mathrm{VM}_i \; ; \text{saturation bw}\right\} \qquad (3)$$

While this initial simple model for the switch is clearly less complex than the ones you can find in network simulators, it can be used to derive very quick decisions within a VM placement algorithm. For example, we can use it to decide if a given VM fits within a given server or, how many VMs can be hosted by a given server architecture.

Based on this initial and simple models, it is necessary to derive from the VMs, the probabilities for changing state and the time how long they stay in a given state, as well as the saturation bandwidth of the virtual switch on a given server system.

III. Realisation Approach

In this section, we have summarised our initial results, in order to determine the limits of different virtual switch configurations and collect initial data on the limits for given server systems. Furthermore, we have used tools that can collect traces from VMs on their communication behaviour in order to develop a traffic model for them.

A. Performance overhead introduced by Open vSwitch VXLAN encapsulation

For the purpose of measuring the performance overhead introduced by network virtualisation, and to establish a plausible baseline for the possible aggregate network throughput of a server equipped with a software-based virtual switch, a test setup was created where both physical non-virtualised networks and virtualised networks using OVS managed *Virtual eXtensible Local Area Network* (VXLAN) tunnels were available. Both network types utilised the same underlying networking hardware, interface configuration and software components, aside from the addition of the tunnelling mechanism itself. Network performance measurements were then conducted from inside virtual machines (instances), which were created and managed through OpenStack [20]. The instances were equipped with two separate tap devices, one connected to the physical network and the other connected to a virtualised network which was transported over the same physical interface.

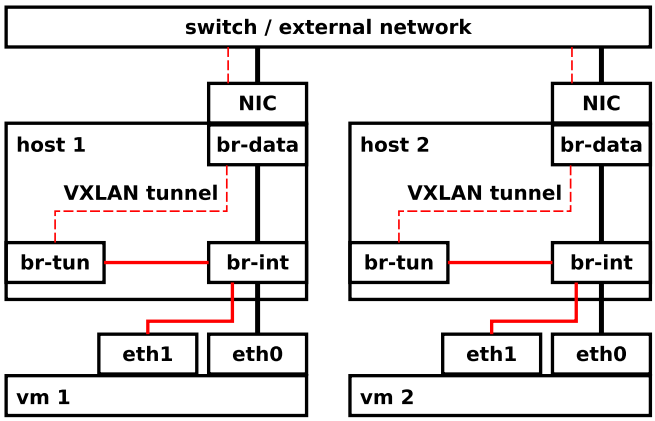


Figure 2. Simplified schematic of network components used for Open vSwitch tunnelling in OpenStack

The OpenStack installation itself consisted of dedicated controller VMs and several physical compute nodes, two of which were used exclusively for the measurements. Each host contained one measurement instance. The physical networking was based on Mellanox ConnectX3-Pro single-port 56GbE QSFP NICs connected to a Mellanox SX1012 12-port 56GbE QSFP switch. The NICs were operated in 56GbE mode with the default *Maximum Transmission Unit* (MTU) of 1500 bytes. Aside from the NICs and the switch, no other physical networking device was located inside the network path for the performance measurements. A more detailed list of specifications for the physical and virtual resources can be found in Appendix A.

In the OpenStack Neutron (networking service) configuration with OVS, which was outlined in Figure 2, the flat external network (solid line from the switch down to eth0) was mapped to an OVS bridge called *br-data*. This bridge was attached to the physical NIC (similar to the external bridge br-ex for network nodes in default OpenStack installations). The virtual tenant networks were attached to the OVS tunnelling bridge *br-tun*, where packets were encapsulated using VXLAN (dotted line) before being transported over the physical NIC. The integration bridge *br-int* connected the various other virtual network components and host-locally isolated different networks using VLANs.

Inside the virtual machines, the network devices for the flat external network and the VXLAN virtual private network were represented as *eth0* and *eth1* respectively. Both devices were configured with a MTU of 1450 bytes to avoid fragmentation on the virtualised private network. Otherwise, fragmentation could occur due to the introduction of additional header bytes by the encapsulation. This MTU configuration also kept the throughput measurements comparable between the two interfaces. To conduct the interface benchmark, the tool IPerf [21] v3.0.11 was used with 4 parallel TCP streams between the client instance and the server instance. Throughput measurements were performed for 300 seconds, to average out slow-start effects and random spikes in the TCP connections.

 ISBN: 978-1-61208-450-3

B. Performance overhead introduced by Open vSwitch VXLAN and GRE encapsulation

The approach mentioned in this section, was to compare the tunnelling protocols for bridging Cloud data centres which are located at different places. A conceptual study on tunnelling protocols was performed and from the study, it was found that VXLAN and *Generic Routing Encapsulation* (GRE) are well suited and both can be generated by using Open vSwitch.

The advantages of using VXLAN is that, it provides 16 million VXLAN ID which overcomes the ID limitation of VLAN. By using VXLAN, we can deploy overlay networks in the virtualised data centres with multi-tenant environment and layer 2 network connection through multiple data centres. The *User Datagram Protocol* (UDP) encapsulation inside VXLAN allows each *Local Area Network* (LAN) segment to be extended across layer 3. VXLAN is also cost-effective as it works in virtualised network, and it saves time as we do not need to deploy many hardware devices. However, VXLAN does not have any control plane. So the network does the work of the control plane such as allocating segment ID and multicast. As VXLAN has no significant support for security, additional measures such as adding *Internet Protocol Security* (IPSec) tunnels are necessary.

While GRE tunnels are less complex to configure and are capable of transmitting multicast traffic, the encapsulation on top of TCP/IP come with the known issues of the limited window size for high speed links. By using GRE tunnels, we can easily transmit packets containing incompatible protocols over the intermediary network via encapsulation with compatible protocols. For example, we can transfer *Internet Protocol version 4* (IPv4) packets over a network that only allows *Internet Protocol version 6* (IPv6) packets. But GRE is even less secured than VXLAN and is more complex in the tunnel set-up procedure.

A synthetic and a database use case were considered to evaluate the performance of the tunnelling protocols and the linked software components in the virtual switch testbed. The measurements included the impact of the tunnelling protocols on point-to-point throughput and latency. As synthetic use cases, an IPerf tool-based evaluation and a *File Transfer Protocol* (FTP)-based evaluation were performed to measure throughput and latency. In the FTP-based evaluation, files of different sizes were downloaded from a FTP server to a FTP client through both tunnelling protocols and the downloaded time was captured. In the database use case, Couchbase database was used to measure the network load when a new instance was added to a cluster.

The tests were performed in two different testbeds. The first testbed included two personal computers and a switch with 1 GbE NIC. The second testbed combined two MicroServers with 10 GbE NIC each. Figure 3 depicts the network topology used in both testbeds. Open vSwitch was used to create the VXLAN and GRE tunnels. In both testbeds, the network settings were kept equal to allow comparison of the results. In testbed 1, the MTU of the 1GbE physical NIC was set to 3000, and 9000 for the 10GbE NIC in testbed 2. The detailed specifications for the two testbeds can be found in Appendix B.

To simulate the distance between the Cloud data centres, network latency between the end systems was increased by using Linux kernel mechanisms in both testbeds. An additional delay between 100ms and 20ms was introduced to the network.

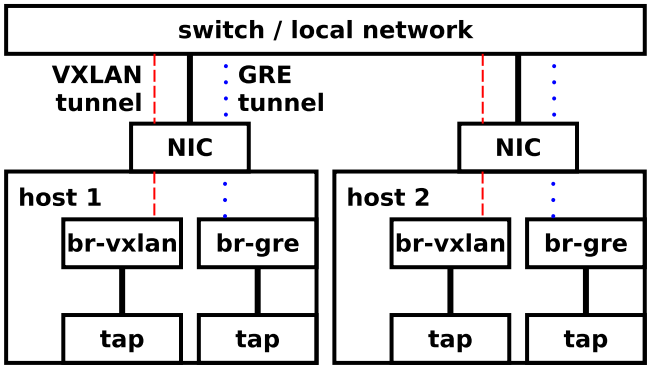


Figure 3. Simplified schematic of network components used for evaluating VXLAN and GRE tunnels in two testbeds.

IV. Results

A. Performance overhead introduced by Open vSwitch VXLAN encapsulation

The results from the measurements using the setup detailed in Section III-A show a significant drop in bandwidth when OVS VXLAN encapsulation was introduced as a network virtualisation mechanism. As seen in Figure 4, throughput in the flat external network averaged $11.289 Gbps$, whereas throughput in the VXLAN-based virtual tenant network only averaged $1.241 Gbps$, roughly one-tenth of the bandwidth in the flat network. Though not detailed in the measurements, the limiting factor in the virtualised network appears to be CPU-bound by the performance of the thread responsible for the packet encapsulation.

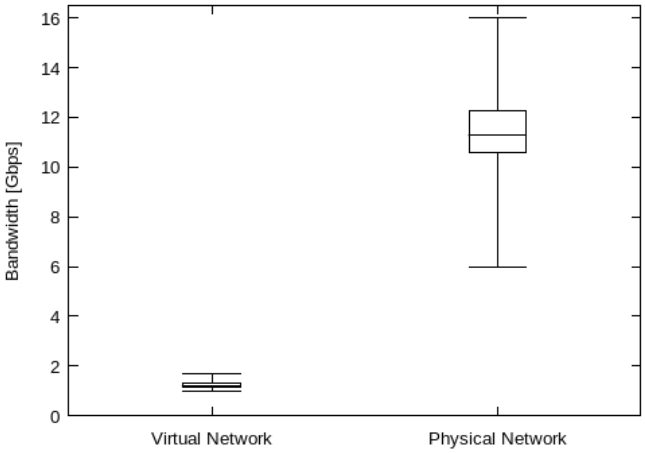


Figure 4. IPerf throughput for VXLAN-based virtual networks and flat physical networks using OpenStack with Neutron OVS networking.

In conclusion, the performance impact of OVS VXLAN encapsulation creates a serious practical limitation for its use in tenant network virtualisation with network interfaces exceeding the 1GbE standard, both in terms of raw throughput and CPU overhead. Therefore, it's use needs to be considered

 ISBN: 978-1-61208-450-3

carefully when instances with network intensive applications are expected.

B. Performance overhead introduced by Open vSwitch VXLAN and GRE encapsulation

Figure 5 represents the IPerf tool-based evaluation and contains a comparison between the bandwidth utilisation of VXLAN and GRE tunnel enabled network.

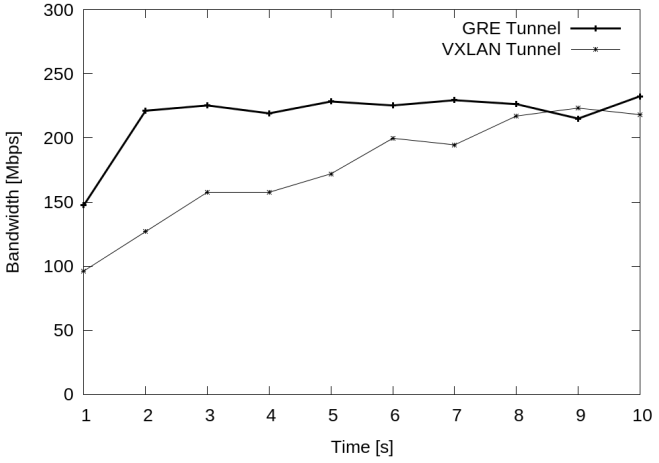


Figure 5. IPerf TCP bandwidth results with 1 stream while downloading a 3GB file through VXLAN and GRE tunnels in testbed 1.

One stream was generated while downloading a 3GB file from the server using FTP. The TCP window size for the testbed was set to the default $85.0KByte$. The measured bandwidths in the VXLAN and GRE tunnel enabled networks were $180465.00Kbits/s$ and $221307.00Kbits/s$ respectively. From the figure, it is seen that the bandwidth utilisation was decreased in both networks due to the additional network delay. But the performance of both tunnels in the network can be seen more clearly. The measured bandwidth was higher in the GRE tunnel.

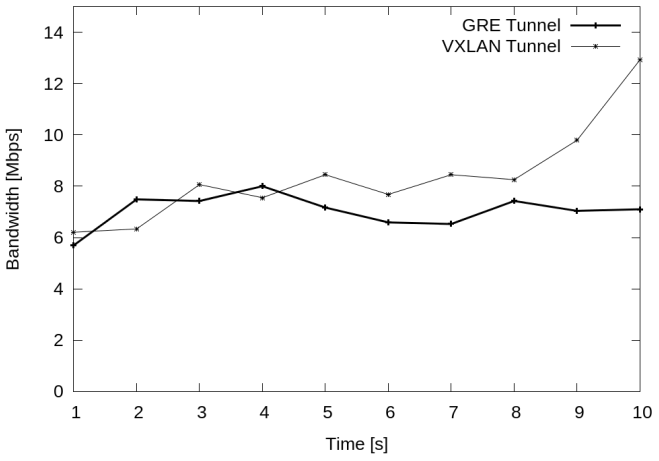


Figure 6. IPerf TCP bandwidth results with 1 stream while downloading a 3GB file through VXLAN and GRE tunnels in testbed 2.

Figure 6, taken from an IPerf tool-based evaluation in the network with 10GbE NIC, represents similar comparison mentioned in Figure 5. The TCP window size in the testbed was kept at the default of $24.5KByte$. The bandwidth utilisation in the VXLAN tunnel enabled network was $8516.00Kbits/s$, which was less than the corresponding value in the network with 1GbE NIC. Similar degradation in bandwidth is visible in the GRE tunnel enabled network. The measured bandwidth in this network was $7159.00Kbits/s$. Both results give clear indication that the network with higher bandwidth was not fully utilised by the tunnels. Also, there are larger fluctuations in the current graph compared to the graph in Figure 5.

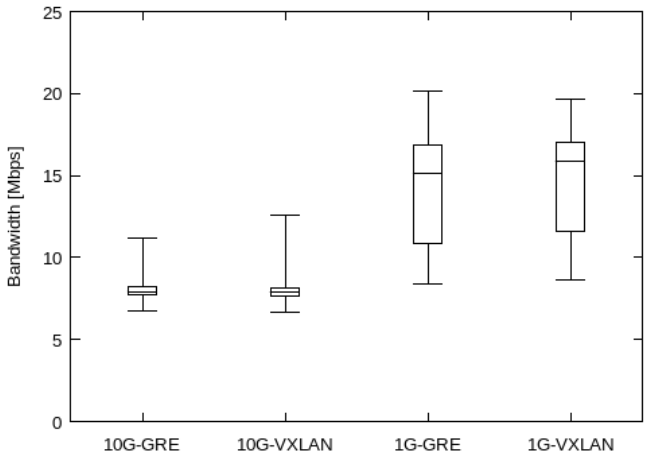


Figure 7. Performance comparison of VXLAN and GRE tunnels while downloading a 10MB file using FTP.

Figures 7 and 8 represent FTP-based evaluations in testbeds 1 and 2. The graphs indicate the impact of VXLAN and GRE tunnels on small file (10MB) and large file (5GB) downloads in both testbeds. Each measurement was repeated 50 times to average out random fluctuations. From the raw data of download time, the bandwidth usage was calculated. The X-Axis represents the testbeds with VXLAN and GRE tunnels. The Y-axis represents the boxplot of the bandwidth usage. From the bottom to the top, the boxplot shows the minimum, the first quartile, the second quartile which is the median, the third quartile and the maximum usage of bandwidth. In Figure 7, we can see that the maximum and minimum bandwidth usage for VXLAN tunnel in the 1st testbed were $19.625Mbps$ and $8.615Mbps$. For GRE tunnel, the values were $20.178Mbps$ and $8.407Mbps$. For the second testbed, the highest and lowest bandwidth usage for VXLAN tunnel were $12.558Mbps$ and $6.674Mpbs$. The corresponding values for GRE tunnel were $11.201Mbps$ and $6.712Mbps$. In comparison, both tunnels showed better performance in testbed 1 than in testbed 2 with respect to the bandwidth.

The effect of the tunnels when downloading large files in both testbeds is depicted in Figure 8. The bandwidth utilisation was higher in the 1GbE NIC network compared to the network with 10GbE NIC. The maximum bandwidth for VXLAN and GRE tunnel in testbed 1 were $36.631Mbps$ and $37.387Mbps$ respectively, whereas for testbed 2, the values were $22.013Mbps$ and $21.969Mbps$. The values for the minimum bandwidth usage followed the same pattern.

For the network with 1GbE NIC, the average performance of the VXLAN tunnel was better than the GRE tunnel. The

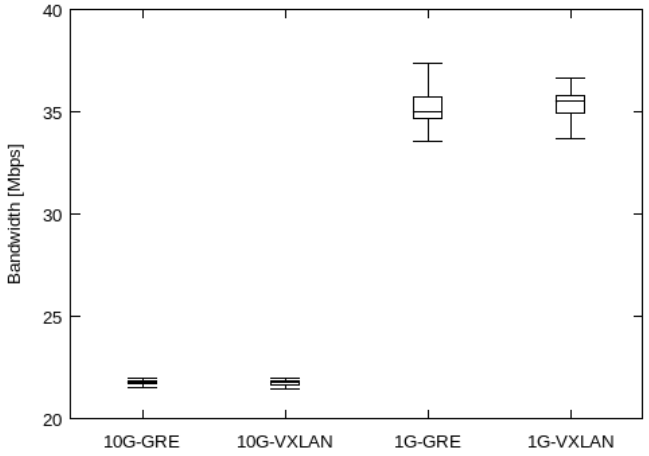


Figure 8. Performance comparison of VXLAN and GRE tunnels while downloading a 5GB file using FTP.

reason for this is the difference in the underlying transport protocols. While VXLAN is based on UDP, the GRE encapsulation operates on top of TCP. For the network with 10GbE NIC, it was detected that Open vSwitch and the weak CPUs of the MicroServers were not compatible with high workloads. Both tunnels delivered degraded performance when there was huge traffic in the network. Also, a high rise in CPU load on the physical machines and an average packet loss of more than 14% were observed during the test in that network. Network delay increased the drop rates and CPU load. Preliminary results indicate that this behaviour might be caused by a non-optimal implementation of Open vSwitch and by the consequences of using virtualisation as it disables hardware features such as TCP offloading. The investigation of this behaviour is still ongoing.

C. Impact on Data Centre Design

As shown in the previous section, depending on the chosen virtual switch solution, the number of VMs that can be placed on a specific server is not determined by CPU cores or memory constraints, but by the maximum throughput that can be achieved with the virtual switch. As the switch performance might be much less than the physical connectivity, even for less communication intensive VMs, the virtual switch performance is the limiting factor.

This leads to the result that, in order to avoid underutilised memory or compute resources, the server design must consider also the virtual switch performance. Furthermore, the chosen approach to place VMs of a customer across the data centre infrastructure has an impact on the performance that can be realised. For Open vSwitch, only small servers (low memory, low core count) with a couple of 1GbE links or, at best a single 10GbE link are cost-effective.

As outlined above, the VM placement algorithms of current Cloud middleware solutions, such as OpenStack, do not consider the communication behaviour of VMs or, constraints introduced by a specific choice of a virtual switch. Therefore, the switch characteristics must be integrated into the placement algorithms e.g., by adding custom filters and removing servers from the list of potential candidates. Figure 9 shows a basic extension that demonstrates the concept of how the placement algorithms could be amended. Still, the basic algorithms only consider average and peak values if the bandwidth demands can be supported by the deployed virtual switch. The next check is validating if the maximum number of tunnels or accumulated bandwidth has been reached.

```
    ...
    while vmList.current()≠ null do
        // check if the bandwidth demand is supported at all
    by the switch type of this server
        if  vmList[i].peakBW ≤ SwitchType.maxTunnel then
 5:         bwSum += vmList[i].averageBW;
            // check if the BW still fits into max BW and max.
    number of tunnels
            if  bwSum > SwitchType.maxPeak
    || vmCount ≥ SwitchType.maxTunnel then
                bwSum -= vmList[i].averageBW;
10:             exit
            end if
        else
            exit
        end if
15:     memoryConsumed += vmList[i].memoryReq;
        coresConsumed += vmList[i].coreReq;
        if currentMemory ≤ memMax
    && currentCoresUsed ≤ coreMax then
            vmCount +=1
20:         vmList.next()
        else
            // we exceeded the available resources...
            exit
        end if
25: end while
    return vmCount
    ...
```

Figure 9. An initial network aware placement decision algorithm.

A more realistic model would work with a switch model that would not only require average and maximum bandwidth demands but would also need the probabilities for certain

bandwidth states and would overlay them. If an additional VM would exceed the available resources, a more realistic decision could be taken based on this model.

The static properties used for a placement decision must be amended with a continuous monitoring of the actual performance data as the network load can be very dynamic. This can then form the basis for a potential VM migration decision. Within the CACTOS project [3], the monitored performance data is fed into an analytics framework and is used to build the basis for an optimisation framework. This optimisation framework not only supports initial placement decisions but also pro-active migration decisions and thus balances between performance benefits and migration costs. The work presented here, is an amendment to the existing system model of CACTOS. It predicts the performance of VMs on different types of compute hosts with a more accurate cut-off bandwidth compared to theoretical peak bandwidth of the installed network interface cards.

V. Conclusion and Outlook

In this paper, we have presented initial results to develop a model that is able to provide decision support for VM placement algorithms in a very short time for selecting an appropriate server to deploy a VM. Furthermore, the models can be used to make a buying decision for a given hardware. In order to determine a good balance between CPU/Memory resources and network capabilities, and to avoid imbalanced server design for Cloud data centres with communication intensive VMs, we can apply the communication behaviour of the models and the virtual switch models.

Further work is needed, in particular to validate if the modelling of VM traffic based on change state probabilities and corresponding rates fits to common workloads. Furthermore, we need to check if the virtual switch model, focused on the saturation bandwidth, is complex enough to properly emulate the behaviour of the different solutions for a wide range of hardware settings. This applies in particular for the tests with Open vSwitch, as the lack of performance of the CPUs in the MicroServer testbed in combination with 10GbE NICs, had significant impact on the overall performance. Also, we observed large variations in the performance for different minor software revisions and kernel versions. Additional comparisons between different virtual switch implementations using different architectural approaches and their effects on available host network resources, are planned for evaluation as well.

Acknowledgement

This research was supported in part by the bwCloud and bwNET100G+ projects funded by the Ministry of Science, Research and the Arts Baden-Wrttemberg (MWK) and the European Unions Seventh Framework Programme funded project CACTOS under grant agreement 610711.

References

[1] C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen et al., "Scinet: lessons learned from building a power-efficient top-20 system and data centre," in Journal of Physics: Conference Series, vol. 256, no. 1. IOP Publishing, 2010, p. 012026.

[2] HPC System "JUSTUS" for computational chemistry. (visited on 2016.01.11). [Online]. Available: http://www.bwhpc-c5.de/wiki/index.php/Hardware_and_Architecture_(bwForCluster_Chemistry)

[3] S. Wesner, H. Groenda, J. Byrne, S. Svorobej, C. Hauser, and J. Domaschka, "Optimised cloud data centre operation supported by simulation," in eChallenges e-2014, 2014 Conference, Oct 2014, pp. 1–9.

[4] "Big data meets high performance computing," Intel corporation, Tech. Rep., 2014.

[5] S. Graf and F. Jülich, "Entwicklung eines werkzeugs zur analyse des os-jitter effekts auf high-performance cluster-systemen," Master's thesis, 2009.

[6] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in INFOCOM, 2012 Proceedings IEEE, March 2012, pp. 2876–2880.

[7] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in INFOCOM, 2013 Proceedings IEEE, April 2013, pp. 355–359.

[8] A. Gupta, D. Milojicic, and L. V. Kalé, "Optimizing vm placement for hpc in the cloud," in Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit. ACM, 2012, pp. 1–6.

[9] Open vSwitch. (visited on 2016.01.11). [Online]. Available: http://openvswitch.org/

[10] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar et al., "The design and implementation of Open vSwitch," in 12th USENIX Symposium on Networked Systems Design and Implementation, 2015.

[11] M. Sarker, "Network infrastructure concept supporting fault tolerance cloud service provision," Master's thesis, Ulm University, Institute of Information Resource Management, 2015.

[12] DPDK. (visited on 2016.01.11). [Online]. Available: http://dpdk.org/

[13] 6WIND, "6WIND Support for Intel Data Plane Development Kit (DPDK)," 2014.

[14] Intel Open Network Platform Server Reference Architecture (Version 1.1), Intel, 2014.

[15] Network Function Virtualization: Intel Data Plane Development Kit vSwitch with Linux Virtualization and Intel Architecture, Intel, 2014.

[16] Lagopus switch. (visited on 2016.01.11). [Online]. Available: https://lagopus.github.io/

[17] Kazuaki Obana, NTT Network Innovation Laboratories, "SDN software switch Lagopus and NFV enabled software node," 2014.

[18] Mellanox OpenStack Solution Reference Architecture, Mellanox Technologies, 2013.

[19] Mellanox CloudX, Mirantis Fuel 5.1 / 5.1.1 Solution Guide, Mellanox Technologies, 2014.

[20] OpenStack Community. OpenStack Open Source Cloud Computing Software. (visited on 2016.01.11). [Online]. Available: https://www.openstack.org/

[21] Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, Kaustubh Prabhu. iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. (visited on 2016.01.11).

 ISBN: 978-1-61208-450-3

APPENDIX A
OPENSTACK ENVIRONMENT SPECIFICATION

OpenStack Hosts	
CPU	2x Intel Xeon E5-2630 v3 @ 2.40Ghz
RAM	16x 16GB DDR4-2133 DIMM REG ECC 2R
Storage	2x SSDs in RAID-1 for OS
Storage	HDD based Ceph with separate SSD cache for VMs
OS	CentOS 7 (current patch level as of the time of the measurements)
NIC	Mellanox ConnectX3-Pro 1-Port 56GbE QSFP, MTU set to 1500
Switch	Mellanox SX1012 12-port 56GbE QSFP switch
Cable	Mellanox QSFP+ 1m passive copper 56GbE capable cable
Kernel	3.10.0-229.11.1.el7.x86_64
vSwitch	Open vSwitch version 2.3.1

OpenStack Instances	
CPU	2 vCores
RAM	4GB
Storage	10GB disk (Nova RBD backend)
OS	CentOS 7 GenericCloud
NIC	eth0: flat external network, MTU 1450
NIC	eth1: VXLAN tenant network, MTU 1450

APPENDIX B
ENCAPSULATION ENVIRONMENT SPECIFICATION

Test Bed 1		
Hardware and software	Number	Specification
Personal computer	2	CPU Intel(R) Core(TM) i54670 CPU @ 3.40GHz, Memory 16GiB, OS Ubuntu 14.04, Kernel version 3.14.27-031427-generic (PC1), 3.13.0-44-generic (PC2)
Switch with 1GbE NIC	1	NETGEAR, Series ProSafe, Model GS108v3, Network interface RJ-45 connector for 10BASE-T, 100BASE-T, or 1000BASE-T Ethernet interface
Ethernet cable	1	Logilink U/UTP CAT6 24AWGX4P PATCH ISO/IEC 11801 and EN 50173 VERIFIED
Open vSwitch	1	version 2.0.2

Test Bed 2		
Hardware and software	Number	Specification
MicroServer	2	CPU Intel(R) Celeron(R) CPU G1610T @ 2.30GHz, Memory 2GiB, OS CentOS 7, Kernel version 3.10.0-123.20.1.el7.x86_64
10GbE NIC	2	HP Ethernet 10Gb 2-port 530SFP+ Adapter, Network Processor QLogic 57810S chipset, Data Rate Two ports, each at 20 Gbps full duplex; 40 Gbps aggregate full duplex theoretical bandwidth.
10GbE cable	2	HP X242 SFP+ SFP+ 3 m Direct Attach Cable (J9283B), Length 10 ft. (3 m)
Open vSwitch	1	version 2.3.1

 ISBN: 978-1-61208-450-3

Experimental Analysis on Performance Anomaly for Download Data Transfer at IEEE 802.11n Wireless LAN

Yoshiki Hashimoto, Masataka Nomoto, Celimuge Wu, Satoshi Ohzahata, and Toshihiko Kato
Graduate School of Information Systems
University of Electro-Communications
Tokyo, Japan
e-mail: hys3224@net.is.uec.ac.jp, noch@net.is.uec.ac.jp, clmg@is.uec.ac.jp, ohzahata@is.uec.ac.jp, kato@is.uec.ac.jp

***Abstract*—It is reported that, even in IEEE 802.11n wireless local area network (WLAN), the performance anomaly occurs which reduces the throughput of all stations when there are some stations communicating with low data rate. But, the previous paper uses a manually configured environment in the Transmission Control Protocol (TCP) uplink performance evaluation. In this paper, we show the results of experiments indicating the performance anomaly at downlink User Datagram Protocol (UDP) and TCP data transfer over 802.11n WLAN. In the experiment, we adopted the actual parameter settings in the stations and the access point, and carefully examined the relationship of the frame aggregation, the queue management at the access point, UDP traffic load, and TCP congestion window size with the performance anomaly. We show that a phenomenon like the performance definitely occurs in both the UDP and TCP data transfer, but the reasons seem to be different for each of them.**

Keywords- WLAN; IEEE802.11n; Performance Anomaly, Queue Management.

I. INTRODUCTION

Resulting from the wide deployment of WLANs based on IEEE 802.11 standard [1], a varieties of WLAN environments including home, office and public hot spots are used by a lot of terminals such as smart phones, tablets and notebooks. When a number of stations access to one WLAN access point, they suffer from several kinds of performance problems. The *performance anomaly* [2][3] is a typical one among those problems.

It is a problem such that: when some stations are located far from their access point and others are near it, the performance of the near stations is degraded to that of far located stations. This is caused by the following two reasons. First, the IEEE 802.11 WLAN is based on the carrier sense multiple access with collision avoidance (CSMA/CA) principle, which tries to assign fair chances to send data frames among all the stations. The other reason is that 802.11 WLAN provides multiple Media Access Control (MAC) level data rates. So, a station with low bit rate captures the channel for a long time, and it penalizes other hosts with higher data rates.

The IEEE 802.11n [1] WLAN introduces several enhancements to the legacy standards such as 802.11a, b and g. Among them, supporting higher data rates (e.g., 150 Mbps), the Aggregated MAC Protocol Data Unit (A-MPDU) and the Block Acknowledgment mechanism provide high performance data transfer.

In spite of the drastic improvement of the data transfer performance, 802.11n standard does not resolve the performance anomaly problem. Abu-Sharkh and Abdelhadi [4] reported that the performance anomaly still exists in 802.11n WLAN. The report [4] describes the results of experiment where four wireless stations perform uplink TCP data transfer. Among the stations, three are located near the access point and one is located far from it. As a result, the performance of near three stations is affected by the far station. The result also shows that the aggregation is closely related with the performance. If the aggregation is used for both near and far stations, the throughput of near stations becomes the same as that of far station. If the aggregation (four MPDUs in an A-MPDU) is used only for near stations, the throughput is four times higher for near stations than far stations.

Although the report [4] mentions the performance anomaly in 802.11n, it uses a manually configured experimental environment, such as a controlled aggregation scheme. In this paper, we show the experimental results of performance anomaly for downlink data transfer in an actual WLAN environment. The feature of our experiment is as follows.

- Both TCP and UDP data transmissions are examined.
- The aggregation scheme in the access point is used as it is.
- The other parameters such as the queue management scheme at the access point are considered.
- For the UDP data transmission, various traffic loads are examined. For the TCP data transmission, the congestion window size is examined in detail together with MAC level data rate.

The rest of this paper consists of the following sections. Section 2 shows the experimental settings. Sections 3 and 4 describe the results of the UDP and TCP experiments, respectively. In the end, Section 5 gives the conclusions of this paper.

II. EXPERIMENTAL SETTINGS

Figure 1 shows the configuration of our experiment. Two stations (STAs) conforming to 802.11n with 5GHz band are associated with one access point, which is connected a server through 1Gbps Ethernet. One STA (STA1) is located at a near position to the access point, and the other STA (STA2) is located in various positions in the experiment.

The detailed specifications of STAs and the access point are shown in Table 1. We use commercially available notebooks and access point in the experiment. As for the

 ISBN: 978-1-61208-450-3

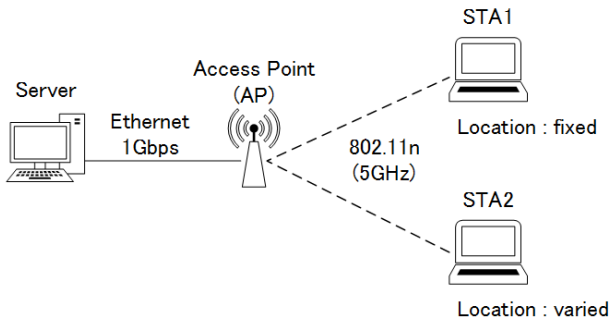


Figure 1. Configuration of experiment

TABLE 1. SPECIFICATIONS OF STAs AND ACCESS POINT (AP)

STA	Manufacturer/Model	DELL Insilon 14
	Operating system	Ubuntu 14.04LTS (kernel 3.13)
AP	Manufacturer/Model	BUFFALO AirStation WZR-HP-AG300H
	Firmware	OpenWRT (BarrierBraker, r444, selfbuild)
	WLAN chip	Atheros AR7161
	WLAN driver	ath9k

access point, we do not use the software originally installed by the vendor but that provided by the OpenWRT project [5]. By using this software, we can obtain the performance metrics in the access point such as the MAC level data rate, the number of A-MPDUs sent, and the number of MPDUs aggregated in an A-MPDU.

We can also configure the queue management schemes used in the access point. The OpenWRT firmware supports the following schemes.

- *FIFO*: A scheme to use one queue to store all packets being sent by the access point, independently of flows the packets belong to.
- *CoDel* [6]: An active queue management scheme designed to resolve the Bufferbloat problem [7]. It uses packet-sojourn time in a queue as a control parameter, and drops a packet in the situation that packets stay in the queue too long.
- *Stochastic Fare Queueing* (*SFQ*): A scheme to provide a separate queue for packets of an individual flow. When sending packets, each queue is examined in the round robin scheduling, which avoids a large delay of packets against an aggressive flow.
- *FQ_CoDel*: A scheme which combines SFQ and CoDel. A queue is prepared for an individual flow and the delay within one queue is controlled by CoDel scheme. In OpenWRT, FQ_CoDel is the default queue management scheme.

In the experiment, we used all those queueing management schemes for the performance evaluation.

The access point uses two streams in the spatial division multiplexing with each channel using 60 MHz bandwidth, and as a result, the data rate ranges from 6.5 Mbps to 300 Mbps.

In the experiment, data is transmitted from the server to two STAs through the access point (downlink data transfer in the WLAN). The server uses *iperf* tool [8] to generate UDP and TCP data flows and to measure their performance such as throughput. As for parameter settings for UDP and TCP, we used the native ones in the Linux operating system. Specifically, the TCP version is CUBIC TCP and the TCP small queues mechanism is used.

During the data transmissions, the following performance metrics data are collected for the detailed analysis of the communication;

- packet trace at the server and the STAs, by use of *tcpdump*,
- WLAN related metrics, such as the MAC level data rate, the number of A-MPDUs sent and the number of MPDUs per A-MPDU, from the device driver at the access point and the stations,
- the throughput and packet loss rate in UDP data transmission, by iperf, and
- TCP connection information such as the congestion window size at the server, by use of *tcpprobe* [9].

The experiment was conducted in a building constructed with reinforced concrete. Figure 2 shows the layout inside the building and the positions of network equipment. The thick black line represents the exterior wall of the building and the thin black line represents the interior wall, which is made from wood. The black circles named "AP" and "STA1" correspond to the positions of the access point and STA1, respectively. These are fixed throughout the experiment. The circles named "Position0" through "Position7" represent the positions of STA2. STA2 is located in one of these eight positons in the experiment.

III. RESULTS FOR UDP DATA TRANSMISSION

A. Experimental Scheme

In the experiment evaluating the performance anomaly using UDP data transmission, we changed the UDP traffic load. For STA1, the server generates UDP datagrams at 100 Mbps, and for STA2, UDP datagrams whose traffic load is 10, 40, 70 and 100 Mbps are generated by the server. In the traffic,

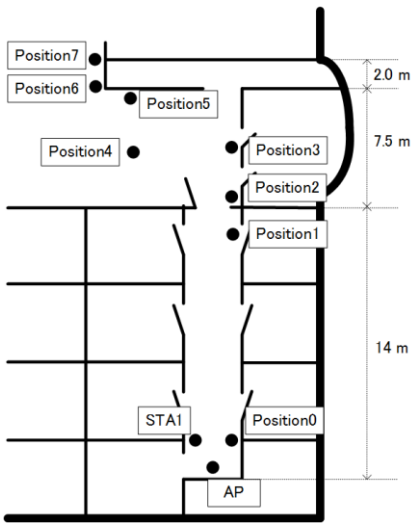


Figure 2. Layout inside building and position of equipment

the size of UDP datagram user data is set to 1472 bytes, which makes the size of IP datagrams 1500 byte.

Eight positions for STA2 and four queue management schemes at the access point are examined. The UDP traffic is generated for 60 seconds in one experiment run, and three runs are examined for one parameter setting.

B. Results

Figure 3 shows the relationship between the position of STA2 and the MAC level data rate (the average during one experiment run) in STA1 and STA2. STA1 located near the access point keeps high data rate such as 200 Mbps through 300 Mbps. On the contrary, the data rate of STA2 decreases along with its position being far away from the access point. More specifically, the data rate of STA2 remains a similar value for Position3 through Position6, and decreases at Position7. The data rate at Position7 has a larger variance than that of the other positions. Figure 3 is the result when the FIFO queue management scheme is used at the access point. The cases when the other schemes are used showed similar results.

In the experiment setup described above, we measured the throughput and the number of MPDUs per A-MPDU for STA1 and STA2, by changing the UDP traffic load of STA2, the position of STA2, and the queue management schemes in the access point. The measured value is the average through three experiment run.

As for the queue management schemes, the results for FIFO and CoDel, and that for FQ_CoDel and SFQ showed similar trends, respectively. It is considered that the packet losses invoked by CoDel do not give any influence to the throughput and the MAC level aggregation behavior in the UDP data transmission. In this section, the results with FIFO and FQ_CoDel schemes are given.

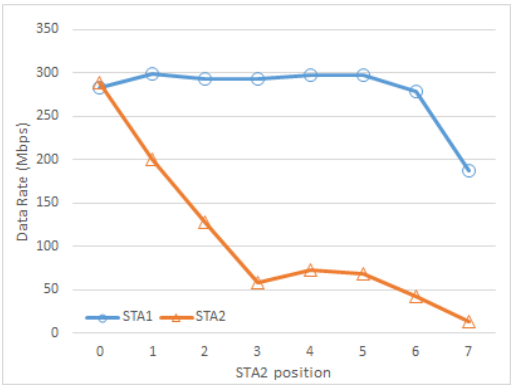
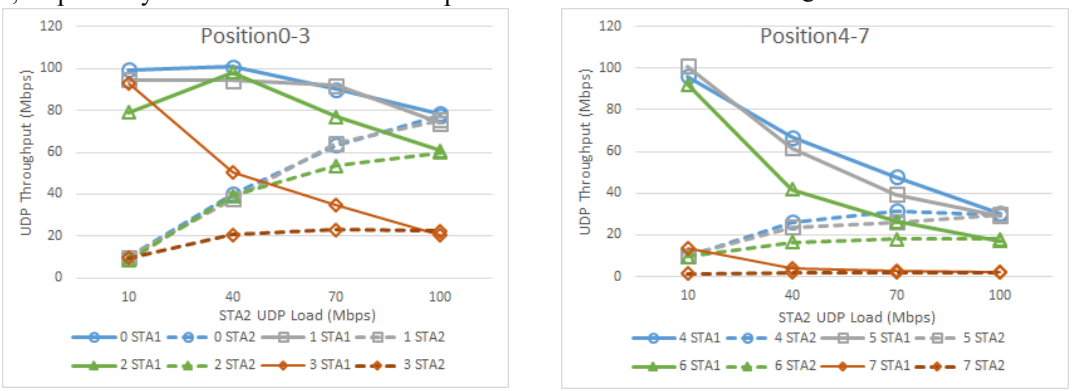


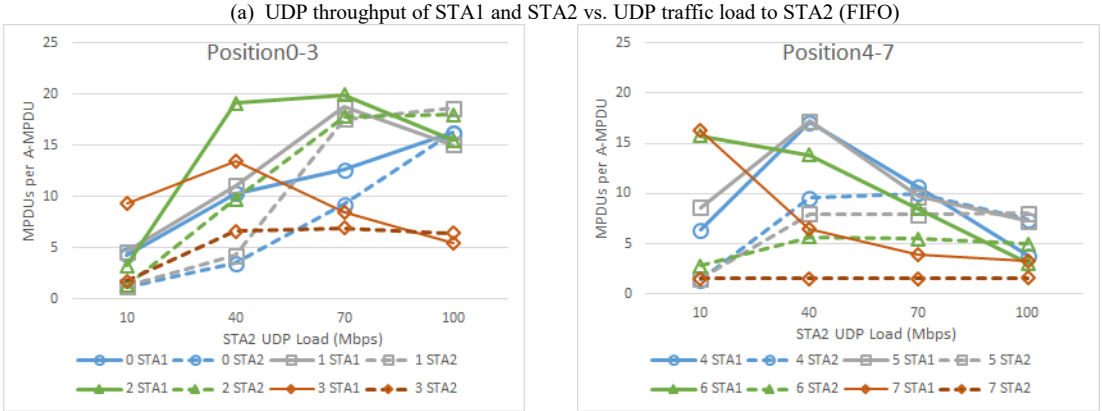
Figure 3. Average MAC level data rate vs. STA2 position (FIFO)

Figure 4 (a) shows the UDP throughput of STA1 and STA2 when the UDP traffic load to STA2 is changed from 10 Mbps to 100 Mbps. (Remember that the UDP traffic load to STA1 is 100 Mbps.) The queue management scheme at the access point is FIFO. The graph shows the results with STA2 located at Position0 through Position7. The solid line is the result of STA1 and the dashed line is for STA2. The color of line discriminates the STA2 position.

When STA2 is located at Position0 where two station can communicate with high MAC level data rate, the UDP



(a) UDP throughput of STA1 and STA2 vs. UDP traffic load to STA2 (FIFO)



(b) Number of MPDUs per A-MPDU vs. UDP traffic load to STA2 (FIFO)

Figure 4. Results for UDP data transmission in FIFO queue management scheme

 ISBN: 978-1-61208-450-3

throughput of STA1 is as high as 80 Mbps. STA2 also provides 80 Mbps throughput at this point. According to locating STA2 far from the access point, the UDP throughput drops. This is because the MAC level data rate used by STA2 becomes low. The drop is larger for higher MAC level data rate. It should be noted that the UDP throughput drop indicates that there several losses of UDP datagrams.

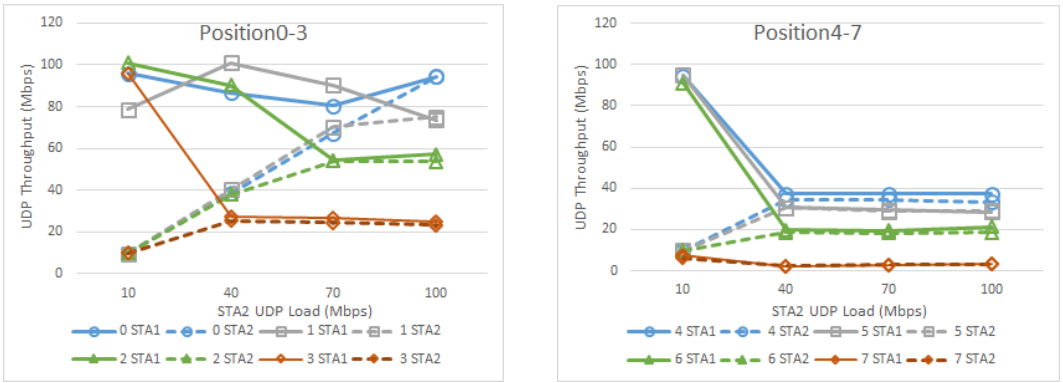
Although the location of STA1 is fixed near the access point and the MAC level data rate STA1 uses is high, the UDP throughput of STA1 becomes lower as STA2 is located far from the access point. When the UDP traffic load to STA2 is 100 Mbps, which is the same as STA1, the UDP throughput is the same for STA1 and STA2. The reason is considered that the low MAC level data rate of STA2 occupies the WLAN channel and the chance of STA1 to transmit data frames will be the same as STA2. This is the performance anomaly.

Figure 4 (b) shows the number of MPDUs per A-MPDU of STA1 and STA2 with the UDP traffic load to STA2 changed from 10 Mbps to 100 Mbps. The queue management scheme in the access point is FIFO. When STA2 is located at Position0, the number of MPDUs per A-MPDU becomes large in both STA1 and STA2, as the UDP traffic load to STA2 increases. This is closely related with the packets queued in the buffer of the WLAN device. At Position0, the MAC level data rate is high for STA1 and STA2. So, while the UDP traffic load to STA2 is low, the buffer of the WLAN device does not contain many data frames to send since the MAC level data rate is high compared with the UDP traffic load. This situation decreases the number of MPDUs aggregated in an A-MPDU. As the UDP traffic load to STA2 increases, the number of MPDUs per A-MPDU increases, and at the STA2 UDP load of 100 Mbps, the numbers for STA1 and STA2 become the same.

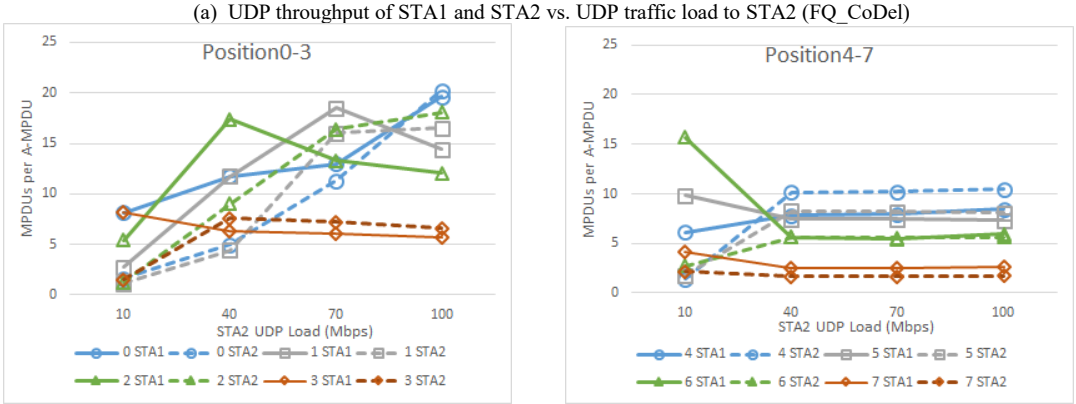
In the case that STA2 is located at Position1 and Position2, the number of MPDUs per A-MPDU of STA1 and STA2 increases along with the UDP traffic load to STA2, until the load is less than and equal to 70 Mbps. When the UDP traffic load to STA2 is more than 70 Mbps, it decreases. As the location of STA2 becomes far from the access point, the number of MPDUs per A-MPDU come to decrease at a lower UDP traffic load to STA2. We infer that this is caused by the aggregating behavior in the WLAN device driver Ath9k [10]. The device driver aggregates MPDUs under the requirement that the transmission time of A-MPDU is below 4 msec. When STA2 is located far from the access point and the MAC level data rate of STA2 decreases, this requirement limits the number of MPDUs aggregated in an A-MPDU in STA2. For STA1, the MAC level transmission rate is equivalently decreased because of the performance anomaly, the number of MPDUs per A-MPDU becomes low.

Figure 5 (a) shows the relationship between the UDP traffic load to STA2, and the UDP throughput of STA1 and STA2, when the access point uses the FQ_CoDel queue management scheme. In this case, similarly to FIFO, the UDP throughput of STA1 becomes lower as the UDP traffic load to STA2 increases. We can say that the performance anomaly also occurs in this experiment.

In contrast to FIFO, where the UDP throughput of STA1 decreases slowly as STA2 is located at Position3 through



(a) UDP throughput of STA1 and STA2 vs. UDP traffic load to STA2 (FQ_CoDel)



(b) Number of MPDUs per A-MPDU vs. UDP traffic load to STA2 (FQ_CoDel)

Figure 5. Results for UDP data transmission in FQ_CoDel queue management scheme

Position7, FQ_CoDel introduces a sharp drop in the UDP throughput of STA1 with UDP traffic load to STA2 larger than 40 Mbps. Figure 5 (b) shows that the number of MPDUs per A-MPDU at STA1 becomes the same as STA2 in this range of UDP traffic load to STA2.

C. Discussions

(1) In the download UDP data transmission over 802.11n WLAN, the performance anomaly surely happens. When the traffic load to a far located station becomes larger than the MAC level data rate which the station uses, it affects the throughput of a near located station. In the experiment, the UDP traffic load to STA2 of 10 Mbps does not invoke the performance anomaly except that STA2 is located at Position7. This is because the MAC level data rate of STA2 is more than 10 Mbps at Position 0 through Position6. The data rate at Position7 is less than 10 Mbps, and this case caused the performance degradation invoked by the performance anomaly.

(2) The UDP throughput also depends on the number of MPDUs aggregated in an A-MPDU. This number depends on the MAC level data rate and the UDP traffic load to the far station. It should be noted that the variation of the number of MPDUs per A-MPDU was large. For example, there was a case where the number of MPDUs changes from 1 to 32. But, the average UDP throughput is determined by the average number of MPDUs per A-MPDU.

(3) The results were affected by the queue management scheme used by the access point. In the case that the access point uses FIFO and CoDel, the data to be sent to both the far and near stations are stored in one queue. So, while the UDP traffic load to the far station is low, more data to the near station are stored in the queue. This makes the number of MPDUs in the near station larger than that of far station. As the UDP traffic load to the far station increases, the number of aggregated MPDUs and the throughput of the far station increases. On the contrary, FQ_CoDel and SFQ maintain separate queues for individual traffic flows. The aggregation is performed queue by queue basis, and so, even in the situation where the performance anomaly occurs, e.g., when the UDP traffic load to STA2 is more than 40 Mbps, the aggregation, and the UDP throughput were similar for the far and near stations.

IV. Results for TCP Data Transmission

A. Experimental Scheme

Similarly with the experiment for UDP data transmission, we measured the performance of TCP data transmission form the server to the stations, by changing the position of STA2 and the queue management scheme in the access point. For each STA2 position and queue scheme, we executed three experiment runs, each of which is 120 second TCP data transfer, and obtained the averages. We measured the MAC level data rate at the access point, the TCP throughput and the number of MPDUs aggregated in an A-MPDU at the receivers (stations), and the TCP congestion window size (cwnd) and the TCP level round trip time (RTT) at the server.

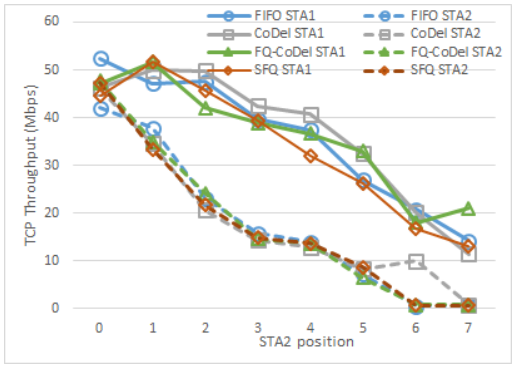
B. Results

As for the MAC level data rate, the similar results were obtained as those in the UDP experiment depicted in Figure 3. STA1 keeps high data rate such as 250 Mbps. The data rate of STA2 decreases as its position is far away from the access point.

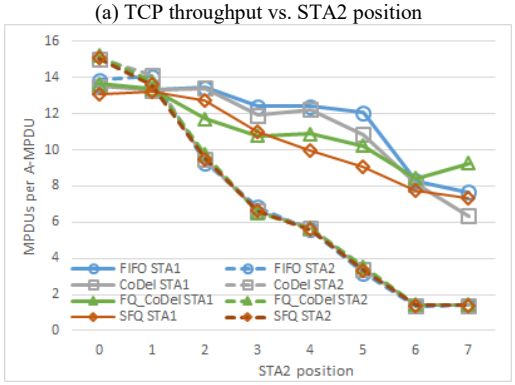
Figure 6 (a) shows relationship between the STA2 position and the average TCP throughput. In this graph, solid lines indicate the results of STA1 and dashed lines indicate those of STA2, and the color of lines correspond to the queue management scheme. The TCP throughput of not only STA2 but also STA1 decreases as the position of STA2 becomes far from the access point. The results seem to be the performance anomaly. In the case of TCP, there was no difference among the queue management schemes in the access point.

Figure 6 (b) shows the relationship between the position of STA2 and the number of MPDUs per A-MPDU. Here, the number of aggregated MPDUs of STA1 and STA2 goes down as the STA2 position becomes far from the access point. Similarly with the TCP throughput, there was no difference among the queue schemes.

Figure 7 (a) shows the average cwnd versus the STA2 position. In this case, the results largely depend on the queue management scheme. In FIFO, the average cwnd is large and it varied from 1 to 900 packets. In SFQ, the queue length for an individual flow is limited to 127 packets, and this in turn limits the cwnd. CoDel and FQ_CoDel drop packets which stay in the queue for a long time, and so the cwnd is suppressed. In all queue schemes, the average cwnd is small when STA2 is located at Position6 and Position7. In this

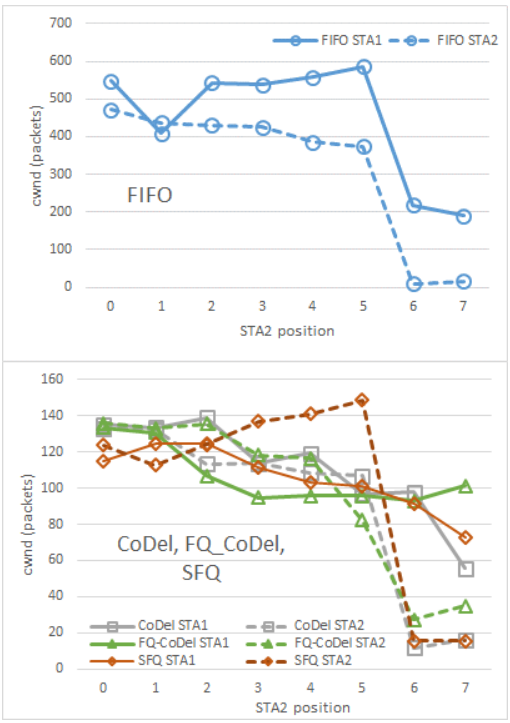


(a) TCP throughput vs. STA2 position

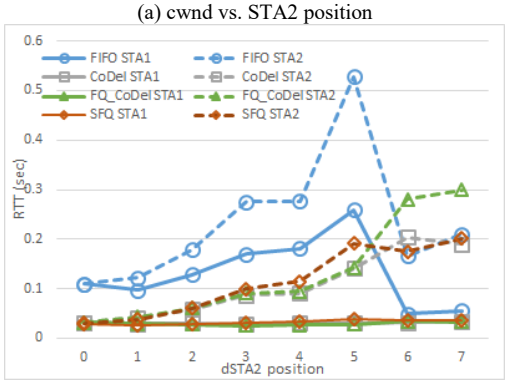


(b) Number of MPDUs per A-MPDU vs. STA2 position

Figure 6. Results of TCP throughput and MPDUs per A-MPDU



(a) cwnd vs. STA2 position



(b) RTT vs. STA2 position

Figure 7. Results of cwnd and TCP level RTT

situation, there are a lot of packet losses in both STA1 and STA2 TCP flows.

Figure 7 (b) shows the average TCP level RTT versus the STA2 position. Here, the results also depend on the queue management scheme. Especially, FIFO has a large RTT compared with the other schemes. In FIFO, both cwnd and RTT are larger than the others, but, since both of them are larger in the similar magnitude, the throughput is also similar with the others.

C. Discussions

(1) In the download TCP data transmission over 802.11n WLAN, the phenomenon similar to the performance anomaly occurs. The throughput of a station located near the access point becomes lower when a far located station communicates. However, the results in Figure 6 (a) show that the throughput of a far located station, STA2, does not exceed the MAC level data rate of STA2. So, it is considered that the reason of the performance anomaly like phenomenon in the TCP data transmission is different from that of the UDP data transmission.

(2) One possible reason is the decrease of cwnd caused by packet losses. As shown in Figure 7 (a), the average cwnd of STA1 decreases as the position of STA2 moves far from the access point. This means that the performance degradation in STA2 invokes the packet losses in the STA1 communication. This invokes the performance anomaly like phenomenon.

V. Conclusions

This paper discussed the performance anomaly of UDP and TCP download data transmissions over IEEE 802.11 WLAN. It showed that the performance anomaly problem surely happens for the UDP data transmission. In the situation where the problem occurs, the number of MPDU aggregation of a near located station is also decreased, and this reinforce the performance degradation. It should be noted, however, that this experiment is done in an artificial condition in which an excessive UDP traffic load is applied. As a result, a large number of packets are lost. In an actual communication, such a packet loss is not acceptable. In other words, for UDP, the performance anomaly problem happens only in the situation where excessive data transfer requests are added.

As for TCP data transmission, the phenomenon similar to the performance anomaly occurs. But, it is considered that the reason is different from that in UDP. The throughput (traffic load) is smaller than the MAC level data rate, and instead, the degradation of the congestion window size caused by packet losses decreases the throughput. More investigation is necessary for the performance analysis of TCP data transmission.

References

[1] IEEE Standard for Information technology: Local and metropolitan area networks Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2012.

[2] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, “Performance Anomaly of 802.11b,” Proc. INFOCOM 2003, vol.2, Mar. 2003, pp.836-843.

[3] M. Abusubaih, “On Performance Anomaly in 802.11 Wireless LANs: Problem and Solution Approaches,” Proc. Next Generation Mobile Applications, Services and Technologies (NGMSAT) 2010, Jul. 2010, pp.208-212.

[4] O. Abu-Sharkh and M. Abdelhadi, “The impact of multi-rate operation on A-MSDU, A-MPDU and block acknowledgment in greenfield IEEE802.11n wireless LANs,” Proc. Wireless Advanced (WiAd), 2011, Jun. 2011, pp. 116-121.

[5] “Open Wrt Wireless Freedom,” https://www.openwrt.org/, retrieved: Jan. 2016.

[6] K. Nichols and V. Jacobson, “Controlling Queue Delay,” ACM Queue, Networks, vol.10, no.5, May 2012, pp. 1-15.

[7] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet,” ACM Queue, Virtualization, vol. 9, no.11, Nov. 2011, pp. 1-15.

[8] iperf, http://iperf.sourceforge.net/, retrieved: Jan. 2016.

[9] Linux foundation: tcpprobe, http://www.linuxfoundation.org /collaborate/workgroups/networking/ tcpprobe, retrieved: Jan. 2016.

[10] ath9k Linux Wireless, http://wireless.kernel.org/en/users/Drivers/ ath9k, retrieved: Jan. 2016.

 ISBN: 978-1-61208-450-3

C_2LP: Modelling Load Propagation and Evaluation through the Cloud Components

Rafael de Souza Mendes*, Rafael Brundo Uriarte†, Carlos Becker Westphall*
*Federal University of Santa Catarina, Florianópolis, Brazil
emails: rafael.mendes@posgrad.ufsc.br, westphal@inf.ufsc.br
†IMT Institute for Advanced Studies Lucca, Italy
email: rafael.uriarte@imtlucca.it

***Abstract*—Due to the scale and dynamism of cloud computing, there is a need for new tools and techniques for its management. This paper proposes an approach to model the load flow in cloud components using double weighted Directed Acyclic Multigraphs. Such model enables the comparison, analysis and simulation of clouds, which assist the cloud management with the evaluation of modifications in the cloud structure and configuration. The existing solutions either do not have mathematical background, which hinders the comparison and production of structural variations in cloud models, or have the mathematical background, but are limited to a specific area (e.g. energy-efficiency), which does not provide support to the dynamic nature of clouds and to the different needs of the managers. In contrast, our model has a formal mathematical background and is generic. To this aim, we present its formalisation and algorithms that support the load propagation and the states of *services*, *systems*, *third-parties* providers and resources, such as: *computing*, *storage* and *networking*. To demonstrate the applicability of our solution, we have implemented a software framework for modelling *Infrastructure as a Service*, and conducted numerical experiments with hypothetical loads.**

Keywords-Cloud Computing; Management; Simulation; Multigraph.

I. INTRODUCTION

The management of pooled resources according to high-level policies is a central requirement of the *as a service* model, as fostered by Cloud Computing (CC). Decision making in the context of clouds, where there exist many possible configurations and complex data flows, is challenging and is still an open issue in the field. In order to use the well-established approaches of *decision theory* [1] and *managerial science* [2] for the CC management, it is necessary to employ formal models to represent the *managed elements* and the flow of the service loads. Furthermore, such a model is also required for the evaluation of possible actions, and for the analysis of cloud components and their hierarchy, which are usually carried out by the management of a cloud *operation*. We highlight this lack of formal models based on our previous efforts to develop methods to CC autonomic management [3][4][5] and formalisms based on Service Level Agreement (SLA) [6].

Currently, the existing solutions which provide CC models can be classified into two main groups: general models, usually represented by the simulators; and specific models, devised for a particular field (e.g., energy saving). The former lack a mathematical formalisation that enables comparisons with variations on the modellings. The latter usually have the formal mathematical background but, since they are specific, they do not support reasoning on different management criteria and encompass only cloud elements related to the target area.

To address this gap in the literature, we analyse the domain elements and characteristics to propose the *Cloud Computing Load Propagation* (C_2LP) graph-based model, a formal schema to express the *load flow* through the cloud computing components. Considering that a *load* expresses the amount of work to process services in systems or resources, the modelling of the load flows enables cloud managers to perform several types of analysis about the cloud structure and behaviour. For example, it enables the comparison of different cloud structures, the distinction of load bottlenecks, the quantitative analysis of the load propagation and of the effects of processing a load in a cloud. In more general terms, such solution unifies heterogeneous abstraction levels of managed elements into a single model and can assist the decision-making tasks in processes, such as: *load balance*, *resource allocation*, *scale up/down* and *migrations*. Moreover, simulations performed using our model can be useful to predict the consequences of managerial decisions and external events, as well as the evolution of baseline behaviour.

More specifically, we model the basic components of CC, the *services*, *systems*, *third-party* clouds that implement services and the resources over which system are deployed: *computing*, *storage*, *networking*. Our model is based on Directed Acyclic Multigraphs. This formalism enables the manager to access consolidated analytical mathematical tools to, e.g., measure the components interdependencies, which is used to improve the availability and resource allocation. In order to demonstrate the applicability and advantages of the C_2LP model, we present a use case where our model is used to compare and evaluate different managerial configurations over several quantitative behaviour in load propagation and evaluation.

This article is organised as follows. Section II discusses the existing cloud models, the works that inspired the definition of this model and the background information necessary for the appreciation of this work. In Section III, we present an overview of the model, formalise it, the propagation algorithm, and the evaluation process. Section IV describes the implementation and the analysis performed on a use case. Finally, in Section V, we discuss the limitations and the future directions for the research.

II. RELATED WORKS

This section presents the related works that propose models to describe and simulate clouds. We have analysed them from a *cloud provider management perspective*, considering their

 ISBN: 978-1-61208-450-3

capacity to: *express general* cloud models, define *components* in distinct abstraction levels; *compare* structures; *simulate* behaviours and provide *formal* specifications with mathematical background. Table I summarises this comparison.

We grouped the proposals into two classes: *general* and *specific*. General models are usually associated with simulators and are used to evaluate numerous criteria at the same time. On the other hand, specific models are commonly associated with particular criterion evaluation, such as: performance [7], security [8][9], accounting [10][11] or energy [12].

All analysed works of the first group, i.e., CloudSim [13], GreenCloud [14], iCanCloud [15], EMUSIM [16] and MDCSim [17], are data-centre oriented, thus requiring extensions to enable the abstraction of important cloud elements, such as services and systems. One exception, the EMUSIM framework, allows the modelling of applications and, from theses models, the estimation of some performance measures but it also depends on data-centre elements, such as: *virtual machines* and *physical machines*. The limitation of the existing generic solutions to deal with abstract elements and their lack of mathematical background hinders the comparison and production of structural variations in cloud modellings, which, in turn, limits their capacity to test the performance of managerial methods.

On the other hand, the solutions in the second group, i.e., the ones specifically devised for an area, present in-depth analysis, based on robust formalisms, such as queue theory [12] [7], probability [8], fuzzy uncertainty [11] and heuristics [10]. However, these models do not fit well in integrated management methods that intend to find optimal configurations considering several criteria of distinct types. Moreover, specific optimisation models usually are sensible to structural changes, having no robustness to support the dynamic nature of the clouds.

The comparison between the related works is presented schematically in Table I, where: the column "Class" specifies if a work is general or specific; "Formalism" evaluates the mathematical background that supports the models; the column "Components" presents the capacity of a model to express cloud components in different abstraction levels; the ability to compare structures is depicted in the column "Comparison"; and, "Simulation" expresses the capacity to perform simulations using the models.

Considering the gap in the existing cloud modelling techniques, our proposal intents to model the load propagation and evaluation functions over a graph to obtain expressiveness, whilst keeping the mathematical background. We opt to model the "load flow" because it is one of the most important information for managerial decisions, such as: load balance, resource allocation, scale up/down and migrations.

TABLE I: COMPARISON BETWEEN RELATED MODELS. ■ REPRESENTS A FEATURE, □ A PARTIALLY COVERED ONE AND - WHEN THE FEATURE IS NOT SUPPORTED.

Model	Class	Formalism	Components	Comparison	Simulation
CloudSim [13]	General	-	■	-	■
GreenCloud [14]	General	-	■	-	■
iCanCloud [15]	General	-	■	-	■
EMUSIM [16]	General	-	■	-	■
MDCSim [17]	General	-	■	-	■
Chang[12]	Specific	■	□	■	□
Püschel [11]	Specific	■	□	■	□
Nesmachnow [10]	Specific	■	□	■	□
Silva. [8]	Specific	■	□	□	□
Vilaplana [7]	Specific	■	□	■	□
C_2LP	General	■	■	■	□

III. MODELLING LOAD FLOW IN CLOUDS

This section discusses the main components of cloud structures and proposes a formal model based on a directed acyclic multigraph, to represent the load flow in clouds.

Subsection III-A presents the structural model and its main components. In Subsection III-B, we formally define the data structures to represent *loads*, *configurations*, *states* and *functions*. Finally, Subsection III-C discusses the computational details of the propagation of the loads and the evaluation of the states for each cloud component.

A. Modelling Clouds with C_2LP

In C_2LP, the structural arrangement of cloud elements is based in a *directed acyclic multigraph* (DAM). The nodes of the graph represent components of the model and can be of four types.

- *Computing*, *Storage* and *Networking* resources, which are the base of any cloud service. These components are always leaf nodes.
- *Systems*, which are abstractions that orchestrate resources that provide and implement services. They can be, e.g., applications and platforms. In the model, systems must be directly linked to at least one of each type of resource: computing, storage and networking. Nevertheless, these resources might be replaced by other systems or third-party services. In such cases, the relationship between the system and the element that represents the resource (i.e., another system or the third-party service) must be explicitly defined using stereotypes (virtual computing, virtual networking or virtual storage).
- *Third-Party Services*, which might represent: (i) resources to system components, when the relation is explicitly annotated with the appropriated stereotype, and (ii) entire systems which provide services and abstract the underlying layers (e.g., email services). The latter models, for example, hybrid clouds or composed services.
- *Services*, which are interfaces between the cloud and the consumers. They are invoked with specification of the consumer's needs and, in this model, they are converted into loads.

Directed edges in our model define to which elements each cloud component can transmit load. Nodes have two main processes: *propagation* of the load; and *evaluation* of the impact of the load in the node itself. Remarkably, the resources components do not propagate load and are the only nodes that actually run the assigned load, while other elements are abstract (e.g., applications, middlewares, platforms and operations systems). Moreover, we consider in the model also the configurations settings of nodes, which impact the propagation and evaluation processes.

The cloud offers services and receives requests from consumers. Requests are then propagated to other nodes using a propagation function that defines to which linked node, the division and in which form the load will be propagated. Since loads might have different forms, we model these relations enabling multiple edges between nodes, which simplifies the understanding of the model. For example, a service transmits

 ISBN: 978-1-61208-450-3

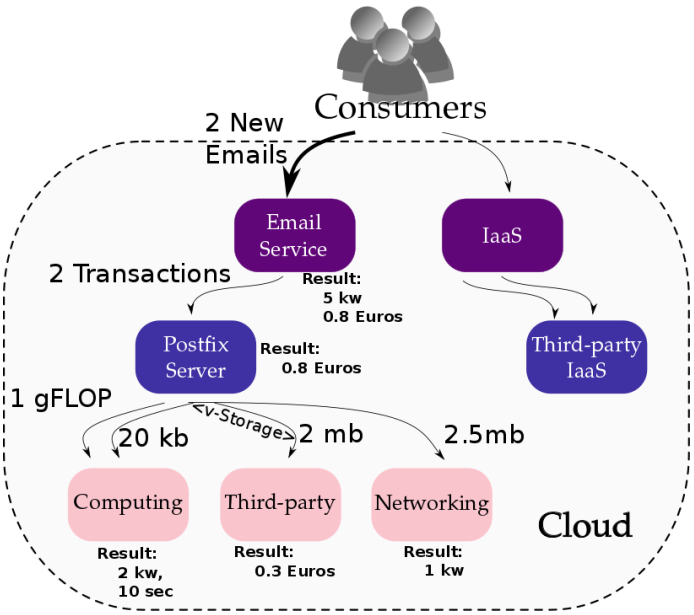


Figure 1: Example of the propagation of loads and the evaluation processes using the C_2LP model.

10 giga FLoating-point Operations Per Second (FLOPS) and 100 giga bytes of data to third-party service. This relation is modelled using two edges, one for each type of load. In case of change in the structure (e.g., a cheaper storage provider) the model can be adjusted simply by removing the storage edge between these nodes and adding it to this new third-party provider.

The evaluation process measures the impact of the load in the resources components and, consequently, on the other nodes of the structure. This process is performed in each node using an evaluation function, which receives three parameters: the configuration of the node, the load defined in the incoming edges and the evaluation of successor nodes. In the case of resource components, the evaluation function uses only the configuration and edge's loads as input.

Figure 1 presents the modelling of a scenario, in which a cloud provides two services: an email and Infrastructure-as-a-Service (IaaS). The IaaS is provided by a third-party cloud. The email service instead, employs a system component to represent a software email server (in this case a Postfix). This component uses local computing and networking and storage from a third-party cloud. The relation (edge) between these components is annotated accordingly.

In the proposed scenario, we exemplify the load propagation with a request from consumers to send 2 new emails using the email service. These 2 emails are converted by the service component into 2 loads of type "transaction" and sent for the email server where, consequently, are converted into another types of load and propagated to the resources linked to the server.

The evaluation process of this scenario uses different metrics in each node and is marked as "result:". For example, in the service level, the load of 2 emails was measured in terms financial cost and energy necessary to complete the request.

B. *Formalisation of the Model*

Formally, in C_2LP model, a cloud C can be expressed as $C = (V, E, \tau^V, \sigma, \Phi, \phi, \Gamma, \gamma, \Gamma', \gamma')$, where:

- V is the set of nodes $V = \{v_1, v_2, \ldots, v_n\}$ of the multigraph, such that every item in V represents one element of the cloud and has one respective node-weight w_v, that usually is a vector of values;
- E is the set of directed edges where $E = \{e_1, e_2, \ldots, e_m\} | e = (v, v')$, that describes the ability of a source node v to transmit a load to node v', such that each e_m also has a respective edge-weight $w_{v,v'}$;
- $\tau^V : V \rightarrow T^V$ is a bijective function which maps the nodes with the respective type, where the set T^V is the set of types of nodes, such that $T^V = \{'computing', 'storage', 'networking', 'system', 'service', 'third\_party'\}$;
- $\sigma : E_{\{\rightarrow system, \rightarrow third-party\}} \rightarrow \{none, vComputing, vStorage, vNetworking\}$ is a function which maps the edges that have systems and third-party services as target with the respective stereotype, characterising the relation between the source element with the target;
- Φ represents the set of propagation functions, where $\Phi = \{f_1, f_2, \ldots, f_v\}$ and ϕ is a bijective function $\phi : V \rightarrow \Phi$ that maps each node for the respective propagation function. Each function in the set Φ is defined as $f_v : \mathbb{N}^n, \mathbb{R}^i \rightarrow \mathbb{R}^o$, where: the set $\mathbb{N}^n$ represents the space where the n-tuple for the configuration is contained; the set $\mathbb{R}^i$ represents the space where the n-tuple of incoming edge-weights is contained; and, $\mathbb{R}^o$ is the space where the n-tuple of the outgoing edge-weights is contained. To simplify the model and the algorithms, we consider that configurations are stored in the node-weight, such that w_v^{conf} represents the configuration part of the node-weight vector.
- Γ is the set of sets that contains the evaluation functions for the leaf nodes, such that there exists one function for each distinct evaluation metric (e.g., energy use, CO2 emission, ...). Then, $\Gamma = \{\Gamma_1, \Gamma_2, \ldots \Gamma_k\}$, such that $\Gamma_k = \{g_{n+1}, g_{n+2}, \ldots, g_m\}$. Each set Γ_k is related to a leaf node $v \in V_{[leaf]}$ through the bijective function $\gamma : V_{[leaf]} \rightarrow \Gamma$. Every g_{n+m} is stored in a distinct position of the node-weight vector of the respective node – representing *a partial state* of v – such that the full new state can be computed through the expression: $w'_v = (c_1, \ldots, c_n, g_{n+1}(c_1, \ldots, c_n, w_v^i), g_{n+2}(c_1, \ldots, c_n, w_v^i), \ldots, g_{n+m}(c_1, \ldots, c_n, w_v^i))$, where: $c_1, \ldots, c_n$ is the n-tuple with the configuration part of the node-weight w_v; w_v^i is the n-tuple with all incoming edge-weights $w_{*,v}$ of v; and w'_v is the new node-weight (full state) for v. The complete evaluation procedure is detailed in Figure 3;
- Γ' is the set of sets that holds the evaluation functions for non-leaf nodes. Therefore, $\Gamma' = \{\Gamma'_1, \Gamma'_2, \ldots, \Gamma'_l\}$, such that each set $\Gamma'_l = \{g'_{n+1}, g'_{n+2}, \ldots, g'_m\}$ contains the evaluation functions g'_{n+m}. Every Γ'_l is associated with a non-leaf node v through the bijective function $\gamma' : V_{non-leaf} \rightarrow \Gamma'$. Since the result of each function g'_{n+m} is stored in a distinct position of w'_v, it represents a partial state of the respective node v. A new full state of non-leaf nodes can be computed through the expression: $w'_v = (c_1, \ldots, c_n,$

1: **procedure** BREADHTFIRSTPROPAGATION(C,W^V,W^E) ▷ Requires a cloud model $C = (V, E, \tau^V, \sigma, \Phi, \phi)$, the set of node-weights $W^V | \forall v \in V \wedge \exists! w_v \in W^V$ and the set of edge-weights $W^E | \forall e_{v,v'} \in E \wedge \exists! w_{v,v'} \in W^E$.
2: $queue \leftarrow \varnothing$
3: $enqueue(*)$
4: **repeat**
5: $v \leftarrow dequeue()$
6: **for each** $u \in successorSet(v)$ **do**
7: $enqueue(u)$
8: **end for** ▷ enqueues the sucessor of each node
9: $f_v \leftarrow \phi(v)$
10: $w_v^{conf} \leftarrow configurationPart(w_v)$ ▷ gets the config. part of the node-weight (state).
11: $w_v^i \leftarrow (w_{1,v}, w_{2,v}, \ldots, w_{u,v})$ ▷ builds the incoming edge-weights in a tuple w_v^i.
12: $w_v^o \leftarrow f_v(w_v^{conf}, w_v^i)$ ▷ w_v^o contains the result of the propagation function.
13: **for each** $w_{v,u} \in w_v^o$ **do**
14: $W^E \leftarrow W^E \oplus w_{v,u}$ ▷ replaces the old value of $w_{v,u}$.
15: **end for**▷ assign the values for the outgoing edges of v.
16: **until** $queue \neq \varnothing$
return W^E
17: **end procedure**

Figure 2: Breadth-first algorithm used for the load propagation.

$g'_{n+1}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$, $g'_{n+2}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$, $\ldots, g'_{n+m}(c_1, \ldots, c_n, w_v^i, w'_{u_v})$); where w'_v is the new node-weight of v, $c_1, \ldots, c_n$ is the n-tuple with the configuration part w_v^{conf} of the node-weight, w_v^i is the n-tuple with the incoming edge-weights $e_{*,v}$ of v, and w'_{u_v} is a tuple which puts together all node-weights of the successors of v (see Figure 3 for details).

The main objective of these formalisms is to specify the data structures that support a model validation, the load propagation, and elements evaluations. The details of each procedure concerned with propagation and evaluations are described in Subsection III-C.

C. Details on the Propagation and Evaluation

The load propagation consists in a top-down process that uses the *breadth-first* approach. In a breadth-first algorithm, the loads are propagated to the neighbour nodes before moving to the next level nodes. In the specific case on C_2LP the algorithm starts from the loads on the services, corresponding to the requests received from consumers.

The propagation process uses a queue with the service nodes (the roots of the graph). Then, a node v is picked from this queue and all its children are placed into the queue. Afterwards, a function $f_v = \phi(v)$ is executed to distribute the load, that is, to define all edge-weights for the outgoing edges of v. This procedure is repeated while the queue is not empty. The well defined method is detailed in Figure 2.

When the load is propagated to resources components (leaf nodes), they execute the load. This execution requires power and resources and can be evaluated in several forms. For example, *energy (kw)*, *performance*, *availability*, *accounting*, *security*, *CO_2 emissions* and other cloud specific feature units. This evaluation process takes every function $g_{n+m} \in \Gamma_k$ in order and computes each partial states, storing them into a position of the new node-weight w'_v. A finer description can be defined as: $w'_v = (w_v^{conf}, g_{n+1}(w_v^{conf}, w_v^i), \ldots, g_{n+m}(w_v^{conf}, w_v^i)$, such that w'_v represents the *a posteriori* state for the node v, w_v^{conf} are the configurations (*a priori* state) of v, w_v^i are the incoming edge-weights of v, and $g_{n+m} \in \gamma(v)$ are the evaluation functions associated with the node.

The evaluations also include the non-leaf nodes since the load also passes through them and it is useful, e.g., to understand the load distribution and to identify bottlenecks. In the case of non-leaf nodes, the evaluation requires also the evaluation results of the bottom nodes. Therefore, this process is performed from the leaves to the roots using a *depth-first* approach.

Resources evaluation occurs according to several models that convert the configuration, loads and the results of other evaluations into node-weights.

A non-leaf node receives the tuples $(config, loads, children\_states)$, and evaluates by the processing of all $g'_{n+m} \in \gamma'(v)$ functions. A representation of this process can be described as: $w'_v = (w_v^{conf}, g'_{n+1}(w_v^{conf}, w_v^i, w'_{u_v}), \ldots, g'_{n+m}(w_v^{conf}, w_v^i, w'_{u_v})$, such that w'_v represents the new node-weight (*a posteriori* state) for the node v, w_v^{conf} are the configuration part (*a priori* state) of node-weight into v, w_v^i represent the incoming edge-weights of v, w'_{u_v} are the computed node-weights of the successors of v, and $g'_{n+m} \in \gamma'(v)$ are the evaluation functions associated with the node.

The complete evaluation process is detailed in Figure 3, where a stack is used to perform a depth-first computation. The first non-visited child of a current node is placed into the stack and will be used as current node. When all children of a node are evaluated, then the node is evaluated. If the node is a leaf node the g functions are used to compute the evaluations, otherwise, the g' functions are used instead.

IV. EXPERIMENTS AND RESULTS

This section presents numerical experiments with the C_2LP model, based on a *service* modelling. These experiments serve to: *test* the applicability of the model; present a concrete *example* of modelling; and, *demonstrate* the model capacity for *quantitative behaviours generation* combining variations of propagation and evaluation functions.

To perform these experiments, we have implemented a use case using our model. This use case exemplifies the model's usage and serves to test its feasibility. The example of model's usage was made using hypothetical functions, since its objective is to prove the generation of simulations, the propagation and the evaluation. Nevertheless, our model can be used for modelling real-world clouds, provided that the propagation and evaluation functions are adjusted to the cloud instance.

As use case, we defined a *IaaS service* where consumers perform five operation: *deploy VM*, *undeploy VM*, *start VM*, *stop VM*, and *execute* tasks. To meet the demand for these services, we designed a hypothetical cloud infrastructure with which is possible to generate quantitative scenarios of propagation and evaluation – in a combinatorial fashion. Using this hypothetical infrastructure, we have tested some managerial configurations related to load distribution over the cloud elements,

 ISBN: 978-1-61208-450-3

1: **procedure** DEPTHFIRSTEVALUATION(C,W^V,W^E) ▷ The same input described in Figure 2.
2: $\beta \leftarrow \varnothing$ ▷ initializes the set of visited nodes.
3: $stack \leftarrow \varnothing$ ▷ initializes the stack.
4: $push(*)$ ▷ starts from the hypothetical node.
5: **while** $stack \neq \varnothing$ **do**
6: $v \leftarrow peek()$ ▷ gets a node without to remove it.
7: **for each** $u \in successorSet(v)$ **do**
8: **if** $u \notin \beta$ **then**
9: $push(u)$
10: **continue while**
11: **end if**
12: **end for** ▷ if the **for** loop ends, all successors have been evaluated.
13: $w^{conf} \leftarrow configurationPart(w_v)$ ▷ gets the config. part for v.
14: $w_v^i \leftarrow (w_1, w_2, \ldots, w_{u,v})$ ▷ builds the n-tuple with the incomings of v.
15: **if** $isLeaf(v)$ **then**
16: $w'_v \leftarrow (w_v^{conf}, g_{n+1}(w_v^{conf}, w_v^i), \ldots, g_{n+m}(w_v^{conf}, w_v^i)), \forall g_{n+m} \in \gamma(v)$ ▷ computes the partial states and builds the new node-weight.
17: **else**
18: $w'_{u_v} \leftarrow (w'_{u_1}, w'_{u_2}, \ldots, w'_{u_o})$ ▷ builds the computed node-weights for all $u | \exists e_{v,u} \in E$.
19: $w'_v \leftarrow (w_v^{conf}, g'_{n+1}(w_v^{conf}, w_v^i, w'_{u_v}), \ldots, g'_{n+m}(w_v^{conf}, w_v^i, w'_{u_v})), \forall g'_{n+m} \in \gamma'(v)$ ▷ computes the partial states and builds the new node-weight.
20: **end if**
21: $W^V \leftarrow W^V \oplus w'_v$ ▷ replaces the old state of v into the node-weights.
22: **if** $v \notin \beta$ **then**
23: $\beta \leftarrow \beta \cup v$
24: **end if** ▷ puts v in the visited set if it is not there.
25: $v \leftarrow pop()$ ▷ gets and removes v from the stack.
26: **end while**
return W^V
27: **end procedure**

Figure 3: Depth-first algorithm to evaluate in specific metrics the impact of the load in each node.

in order to evaluate the average utility for all quantitative scenarios. At the end, the configurations which achieve the best average utility for all quantitative scenarios were highlighted, depicting the ability of the model to simulate configuration consequences for the purpose of selecting configurations.

A. Use Case Modelling

To deal with the consumers' loads (deploy, undeploy, start, stop and execute), the infrastructure manages: the *service interface*; systems, such as *load balancers*, *cloud managers* and *cloud platforms*; and resources, such as *servers*, *storages* and *physical networks*. All operations invoked by consumers represent an incoming load on the service interface, which is propagated to resources. In the resources the loads are evaluated to provide measures about *performance*, *availability*, *accounting*, *security* and *CO_2 emissions*. These evaluations are then executed also for systems and, at the end, for the service interfaces.

The modelling of the use case was devised considering 21 components: 1 service, 9 systems, and 11 resources. The services represent the interface with customers. In this use case, the systems are: a *load balancer*; two *cloud manager* systems; and six *cloud platforms*. Also, between the resources there are: 8 physical computing servers (6 work servers and 2 managerial), 2 storages (1 work storage and 1 managerial), and 1 physical network. A detailed list of components is presented in Appendix I.

Regarding the edges and loads, each consumer's operation is modelled as an incoming edge in a *service interface node* – with the respective loads in the edge-weights. The service node forwards the loads for a *load balancer* system, where the propagation function decides to which *cloud manager* the load will be sent, whereas the *manager servers*, the *manager storage* and the *physical network* receive the loads by its operation. In the cloud mangers, the propagation function must decide to which *cloud platform* the loads will be sent and, at the same time, generate loads for the managerial resources. The cloud platform system effectively converts its loads into simple resource loads when uses the *work server*, *work storage* and *physical network*. The complete relation of load propagation paths is presented in Appendix I, where an element at the left side of an arrow can propagate loads for an element at the right. Furthermore, a graphical representation of these tables, which depicts the graph as a whole, is also presented in Appendix I.

Besides the node and the edges, the use case model required the definition of: • 4 types of propagation functions – one for the service and tree for each type of system; • 6 types of leaf evaluation functions – two specific performance evaluations, one for computing resources and another for storage and networking; plus, four common evaluation functions (availability, accounting, security and CO_2 emissions) for each type of resource; • 5 types of non-leaf evaluations functions.

We have modelled the possible combinations to distribute the loads $\{1\text{-deployVM}, 2\text{-undeployVM}, 3\text{-startVM}, 4\text{-stopVM}, 5\text{-compute}\}$ as a partition set problem [18], resulting in 52 distinct possibilities of load propagation. Also, we introduced 2 possible configurations into each evaluation function for leaf nodes. These configurations are related to the choice of constants into the function. For example, the performance of a computing resource depends on its capacity, that can be: $a = 50GFLOPs$ or $b = 70GFLOPs$. Considering 5 distinct evaluation functions over 11 leaf nodes, we have got $(2^5)^{11} = 2^{55}$ possible distinct configurations to test.

B. Evaluations

The numerical experiments were performed running the propagation procedure, followed by the evaluation of every simulation. For each possible propagation, we tested and summarized the 2^{55} configurations for evaluation functions. Then, we analysed the average time (p, in seconds), average availability (av, in %), average accounting (ac, in currency units), average security (s, in % of risk of data exposition), and average of CO_2 emissions (c, in grammes). Each value was normalised according to the average for all propagations, tested and summarised in a global utility function, described in (1) – where the overlined variables represent the normalised values.

Such results enable cloud managers to choose the best

 ISBN: 978-1-61208-450-3

TABLE II: SUMMARY OF AVERAGE EVALUATIONS FOR EACH CONFIGURATION.

Criteria	Configuration			
Code	11221	11231	11232	**11212**
Time	180.59976	180.5999	180.60004	**180.59991**
Availability	0.9979606	0.99795955	0.9979587	**0.99795926**
Accounting	78.69924	78.69926	78.699234	**78.699265**
Security	0.9979606	0.99795955	0.9979587	**0.99795926**
Emissions	82848.31	82848.14	82848.51	**82848.74**
Utility	1.0526400204	1.0526410547	1.0526477776	**1.0526491889**

scenario according to the priorities of the policy or to provide input for the decision-making process, such as Markov Chains.

$$u = -(\overline{av} + \overline{s} - (\overline{p} + \overline{ac} + \overline{c})) \quad (1)$$

The best four results of the fifty two numerical experiments are presented in Table II in ascending order, where the configuration that achieves the best *average utility* is highlighted in bold. The *code* line in the table represents the propagation configuration, whereas the other lines contain the values obtained for each distinct evaluation type. The last row presents the average utility defined in Equation 1. To represent configuration we have adopted a set partition notation to express the propagation paths, such that each position in the code represents a type of load: 1-*deploy*, 2-*undeploy*, 3-*start*, 4-*stop*, and 5-*compute*. Considering that at leaves of the propagation graph there are 6 cloud platforms, a code 11212 indicates that the loads of type 1,2 and 4 were allocated on cloud platform 1, whereas the loads 3 and 5 ware allocated in the cloud platform 2.

V. CONCLUSION

Several solutions have been proposed to model clouds. However, to the best of our knowledge, none is general and has mathematical formalism at the same time, which are essential characteristics for the consolidation of analytical methods.

In this study, we have presented an approach with these characteristics to model clouds based in *Directed Acyclic Multigraph*, which has the flexibility of general models and the formalism of the specifics. Therefore, C_2LP is a flexible well-formed modelling tool to express flow of loads through the cloud components. This model supports the specification of elements in distinct abstraction levels, the generation of combinatorial variations in a use case modelling and the evaluation of the consequences of different configuration in the load propagation.

We developed a simulation software tool for the modelling of IaaS services and demonstrated the applicability of our approach through a use case. In this use case, we simulated several graph network theoretic analysis, evaluated and compared different configurations and, as a result, supplied the cloud managers with a numeric comparison of cost and benefits of each configuration. These experiments, demonstrated that this tools provides an essential support for the management of cloud.

ACKNOWLEDGEMENT

The present work was done with the support of CNPq agency, through the program Ciência sem Fronteiras (CsF), and the company Eletrosul Centrais Elétricas S.A. – in Brazil. The authors also would like to thank professor Rocco De Nicola and the group of SysMA research unit in Institute for advanced studies Lucca (IMT).

REFERENCES

[1] Itzhak Gilboa, "Theory of Decision under Uncertainty," Cambridge University Press, 2009.

[2] Cliff Ragsdale, "Modeling & Decision Analysis," Thomson, 2008.

[3] Rafael Mendes et al., "Decision-theoretic planning for cloud computing," In ICN 2014, The Thirteenth International Conference on Networks, Iaria, vol. 7, no. 3 & 4, 2014, pages 191–197.

[4] Alexandre A. Flores, Rafael de S. Mendes, Gabriel B. Bräscher, Carlos B. Westphall, and Maria E. Villareal, "Decision-theoretic model to support autonomic cloud computing," In ICN 2015, The Fourteenth International Conference on Networks, Iaria, vol. 8, no. 1 & 2, 2015, pages 218–223.

[5] Rafael Brundo Uriarte, "Supporting Autonomic Management of Clouds: Service-Level-Agreement, Cloud Monitoring and Similarity Learning," PhD thesis, IMT Lucca, 2015.

[6] Rafael Brundo Uriarte, Francesco Tiezzi, and Rocco De Nicola, "SLAC: A formal service-level-agreement language for cloud computing," In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 2014, pages 419–426.

[7] Jordi Vilaplana, Francesc Solsona, and Ivan Teixidó, "A performance model for scalable cloud computing," In 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015), ACS, vol. 163, 2015, pages 51–60.

[8] Paulo F Silva, Carlos B Westphall, and Carla M Westphall, "Model for cloud computing risk analysis," ICN 2015, Iaria, vol. 8, no. 1 & 2, 2015, page 152.

[9] Nada Ahmed and Ajith Abraham, "Modeling cloud computing risk assessment using machine learning," In Afro-European Conference for Industrial Advancement, Springer, 2015, pages 315–325.

[10] Sergio Nesmachnow, Santiago Iturriaga, and Bernabe Dorronsoro, "Efficient heuristics for profit optimization of virtual cloud brokers," Computational Intelligence Magazine, IEEE, vol. 10, no. 1, 2015, pages 33–43.

[11] Tim Püschel, Guido Schryen, Diana Hristova, and Dirk Neumann, "Revenue management for cloud computing providers: Decision models for service admission control under non-probabilistic uncertainty," European Journal of Operational Research, Elsevier, vol. 244, no. 2, 2015, pages 637–647.

[12] Chunling Cheng, Jun Li, and Ying Wang, "An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing," Tsinghua Science and Technology, IEEE, vol. 20, no. 1, 2015, pages 28–39.

[13] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Csar A. F. De Rose, and Rajkumar Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, Wiley Online Library, vol. 41, no. 1, 2011, pages 23–50.

[14] Dzmitry Kliazovich, Pascal Bouvry, and Samee Ullah Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," The Journal of Supercomputing, Springer, 2012, page 1263–1283.

[15] Alberto Núñez et al., "iCanCloud: A flexible and scalable cloud infrastructure simulator," Journal of Grid Computing, Springer, vol. 10, no. 1, 2012, pages 185–209.

[16] Rodrigo N Calheiros, Marco AS Netto, César AF De Rose, and Rajkumar Buyya, "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," Software: Practice and Experience, Wiley Online Library, vol. 43, no. 5, 2013, pages 595–612.

[17] Seung-Hwan Lim, Bikash Sharma, Gunwoo Nam, Eun Kyoung Kim, and Chita R Das, "MDCSim: A multi-tier data center simulation, platform," In Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, IEEE, 2009, pages 1–9.

[18] Toufik Mansour, "Combinatorics of Set Partitions," CRC Press, 2012.

 ISBN: 978-1-61208-450-3

APPENDIX I: IMPLEMENTATION DETAILS

TABLE III: THE CLOUD ELEMENTS – NODES OF THE GRAPH.

CS - computing service	**CP21** - platform 21	**WS12** - work server 12
LB - load balancer	**CP22** - platform 22	**WS13** - work server 13
CM1 - cloud manager 1	**CP23** - platform 23	**WS21** - work server 21
CM2 - cloud manager 2	**MS1** - manager server 1	**WS22** - work server 22
CP11 - platform 11	**MS2** - manager server 2	**WS23** - work server 23
CP12 - platform 12	**MSTO** - manager storage	**WSTO** - work storage
CP13 - platform 13	**WS11** - work server 11	**PN** - physical network

TABLE IV: THE LOAD PROPAGATION RELATIONS – EDGES OF THE GRAPH.

$\xrightarrow{5}$CS	CM1$\xrightarrow{5}$CP11	CP11 $\rightarrow$ WS11	CP21 $\rightarrow$ PN
CS$\xrightarrow{5}$LB	CM1$\xrightarrow{5}$CP12	CP11$\rightarrow$PN	CP21$\rightarrow$WSTO
LB$\xrightarrow{5}$CM1	CM1$\xrightarrow{5}$CP13	CP11$\rightarrow$WSTO	CP22$\rightarrow$W22
LB$\xrightarrow{5}$CM2	CM1 $\rightarrow$ PN	CP12$\rightarrow$WS12	CP22 $\rightarrow$ PN
LB $\rightarrow$ MS1	CM2 $\rightarrow$ MS2	CP12 $\rightarrow$ PN	CP22 $\rightarrow$ WSTO
LB $\rightarrow$ MS2	CM2 $\rightarrow$ MSTO	CP12 $\rightarrow$ WSTO	CP23 $\rightarrow$ W23
LB $\rightarrow$ WSTO	CM2 $\xrightarrow{5}$ CP21	CP13$\rightarrow$ W13	CP23 $\rightarrow$ PN
LB $\rightarrow$ PN	CM2 $\xrightarrow{5}$ CP22	CP13$\rightarrow$ PN	CP23 $\rightarrow$ WSTO
CM1 $\rightarrow$ MS1	CM2 $\xrightarrow{5}$ CP23	CP13 $\rightarrow$ WSTO	
CM1 $\rightarrow$ MSTO	CM2 $\rightarrow$ PN	CP21 $\rightarrow$ W21	

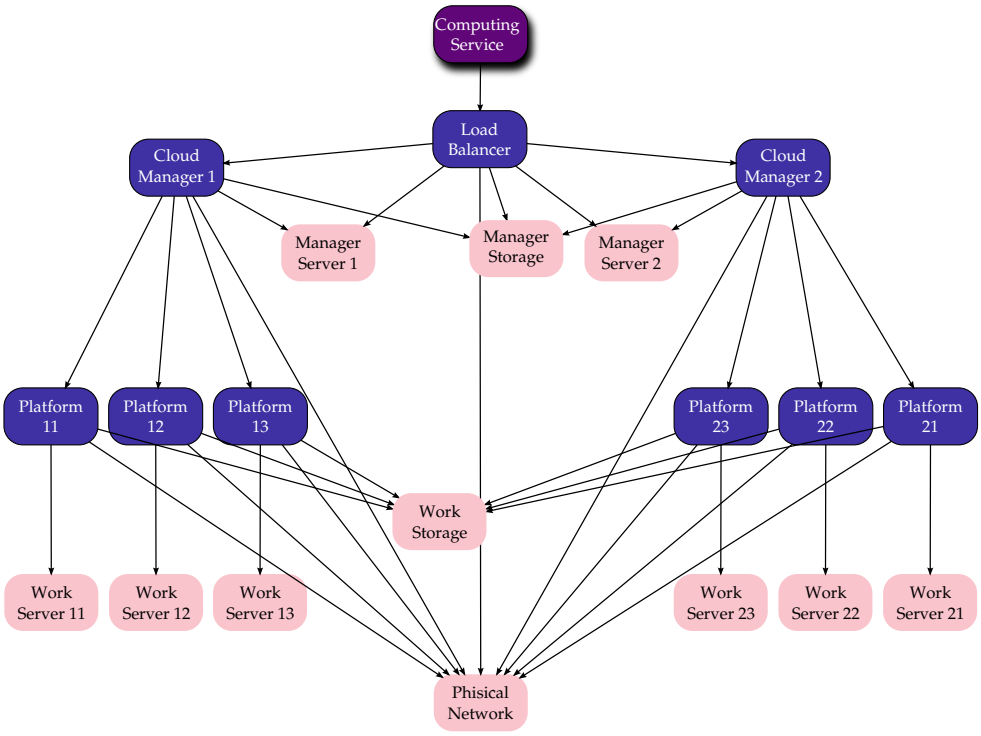


Figure 4: Graphical representation of structural arrangement for the modelling use case.

TABLE V: PROPAGATION FUNCTIONS.

Types	Declarations	Definitions
service	$(w_1, \cdots, w_5) \overset{f^{CS}}{\longmapsto} (w'_1, \cdots, w'_5)$.	w_n is the weight for $\overset{n}{\rightarrow} CS$). w'_n is the weight for $(CS \overset{n}{\rightarrow} LB)$. $w'_n = w_n \vert \forall w'_n \in f^{CS}$.
balancer	$(c_1, \ldots, c_5, w_1, \cdots, w_5) \overset{f^{LB}}{\longmapsto}$. $(w'_1, \ldots, w'_{14})$	$c_n \in \{CM1, CM2\}$, are the configurations which represent the targets of each load $w_n \vert 1 \leqslant n \leqslant 5$. $w'_n = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CM1 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert 1 \leqslant n \leqslant 5$. $w'_{n+5} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CM2 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert 1 \leqslant n \leqslant 5$. $w'_{1 \geqslant n \geqslant 5}$, are the weights in the edges LB$\overset{5}{\rightarrow}$CM1. $w'_{6 \geqslant n \geqslant 10}$, are the weights in the edges LB$\overset{5}{\rightarrow}$CM2. $w'_{11} = 1Gflop$, is the a constant computing load in LB$\rightarrow$MS1. $w'_{12} = 1Gflop$, is the a constant computing load in LB$\rightarrow$MS2. $w'_{13} = 50GB$, is the a constant storage load in LB$\rightarrow$MSTO. $w'_{14} = w_1 + 40$, is the load over LB$\rightarrow$PN, such that w_1 is the VM image size in *GB*, comes from *deploy VM* operation, and 40 is a constant value in *GB* for the another operations.
cloud manager	$(c_1, \ldots, c_5, w_1, \cdots, w_5) \overset{f^{CMn}}{\longmapsto}$ $(w'_1, \ldots, w'_{18})$	$c_n \in \{CPm1, CPm2, CPm3\}$, are the configurations which represent the targets of each load $w_n \vert 1 \leqslant n \leqslant 5$. $w'_n = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm1 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert 1 \leqslant n \leqslant 5$. $w'_{n+5} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm2 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert 1 \leqslant n \leqslant 5$. $w'_{n+10} = \left\{ \begin{array}{cc} w_n & \text{if } c_n = CPm3 \\ 0 & \text{otherwise} \end{array} \right. \Big\vert 1 \leqslant n \leqslant 5$. $w'_{16} = 1Gflop$, is the a constant computing load in CMn$\rightarrow$MSn. $w'_{17} = 50GB$, is the a constant storage load in CMn$\rightarrow$MSTO. $w'_{18} = w_1 + 40$, is the load over CMn$\rightarrow$PN, such that w_1 is the VM image size in *GB*, comes from *deploy VM* operation, and 40 is a constant value in *GB* for the another operations.
cloud platform	$(w_1, \cdots, w_5) \overset{f^{CPnn}}{\longmapsto} (w'_1, w'_2, w'_3)$.	$w_1, \cdots, w_5$, are the main loads come from the service, associatively, w_1 – deploy VM, w_2 – undeploy VM, w_3 – start VM, w_4 – stop VM, and w_5 – compute tasks. w'_1, w'_2 and w'_3 are, respectively, the edge-weight for the arcs CPnn$\rightarrow$WSnn, CPnn$\rightarrow$WSTO and CPnn$\rightarrow$PN, where: $w'_1 = w_1 - w_2 + w_3 - w_4 + w_5$; $w'_2 = w_1 - w_2 + 1MB$; $w'_3 = w_1 + w_3 - w_4 + 1MB$.

TABLE VI: EVALUATION FUNCTIONS FOR LEAF NODES.

Types	Functions
computing specific functions	*performance* (*duration*): $d(load) = \frac{load}{capacity}$, where $load$ is expressed in GFlop, $capacity$ is a constant of 70GFLOPs and d is the total time to resolve the load. *energy increment* (kWh): $energy_{increment}(load)$ here is considered a linear function which returns the amount of energy necessary to process the load above the average consumption of standby state. For computing have been considered $0.001kW$ per GFLOP.
storage and network specific functions	*performance* (*duration*): $d(load) = \frac{load}{capacity}$, where $load$ is expressed in GByte, $capacity$ is a constant of 1GBps and d is the total time to resolve the load. For the networking resources this concept is intuitively associated with the network throughput, however, for storage is necessary to explain that the performance refers to throughput of the data bus. *energy increment* (kW): $energy_{increment}(load)$ for data transmission is assumed as linear, and was here considered $0.001kW$ per GB transferred.
common functions	*availability*: $av(load) = 1 - p_{fault}(d(load))$, where p_{fault} is the probability which a fault occurs during the load processing. Here will be considered a linear naive probability, such that $p_{fault}(d) = d \times 0.01$. *accounting*: $ac(load) = price_{energy} \times energy_{total}$, where $price_{energy}$ is a constant of $0.38US\$/kW$ or $0.58US\$/kW$, depending on node configuration; and $energy_{total} = energy_{increment}(load) + energy_{average}(d(load))$, such that $energy_{average}(d(load)) = d(load) \times 0.1kW$ is the shared energy spent by the cloud by time slot, and $energy_{increment}(load)$ is the increment of energy result of resource usage. *security* (*risk of data exposition*): $s(load) = 1 - p_{exposure}(load)$, where $p_{exposure}(load)$ is the probability that the load processing results in data exposure and $s(load)$ is the trustability of the operation. The $p_{exposure}(load)$ is calculated as 0.001 for each second of operation. *CO_2 emission*: $c = energy_{total} \times 400$, where $energy_{total}$ was defined in the accounting evaluation function and 400 is a constant which represents the grammes of CO_2 per kW.

TABLE VII: EVALUATION FUNCTIONS FOR NON-LEAF NODES.

Types	Declarations	Definitions
performance	maximum duration of loads sent for successor nodes.	$p_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = max(w'_1[p], \ldots, w'_n[p])$, where p_v represents the total time to process the incoming loads, and $w'_n[p]$ represents the specific part of in the node-weight of n successor nodes, regards to the duration to process the loads sent by the node v.
availability	the product of the availability of successor nodes according to the sent loads.	$av_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \prod w'_n[av]$, where av_v represents the total availability of a node v according its dependencies, and $w'_n[av]$ represents the availability part in node-weights of the successors of v, related to the loads sent.
accounting	the sum of costs relative to the sent loads for successor nodes.	$ac_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \sum w'_n[ac]$, where ac_v is the total cost related to v and regards to the loads processed in the successors, and $w'_n[ac]$ is the accounting part of the successors' node-weight.
security	the product of security (regards to data exposition) of successor nodes according to the sent loads.	$s_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \prod w'_n[s]$, where s_v represents the total security measure of a node v, and $w'_n[s]$ represents the security measure part in node-weights of the successors of v, related to the loads sent.
CO_2 emission	the sum of total emissions relative to the loads sent to the successor nodes.	$c_v(w_1, \ldots, w_5, w'_1, \ldots, w'_n) = \sum w'_n[ac]$, where c_v is the total CO_2 emission associated with a node v, and $w'_n[ac]$ is the node-weight part associated with the emissions caused by the loads sent from v.

 ISBN: 978-1-61208-450-3

Chain-of-Trust Packet Marking

Otávio A. S. Carpinteiro, Phyllipe L. Francisco, Pablo M. Oliveira, Edmilson M. Moreira

Research Group on Systems and Computer Engineering
Federal University of Itajubá, 37500–903, Itajubá, MG, Brazil
Emails: {otavio,phyllipe,pablo,edmarmo}@unifei.edu.br

***Abstract*—This paper proposes a new deterministic traceback method — chain-of-trust packet marking (CTPM). CTPM establishes a chain of trust composed of the border routers of the autonomous systems (ASes). It makes use of a new IPv6 extension header — the traceback extension header (TEH) — which extends the datagram size in 168 bytes at most. The TEH contains encrypted marks which trace the path taken by each IPv6 datagram from its origin to the destination. The computational load on the border routers, as well as the network latency generated by CTPM will be measured and evaluated in future studies.**

***Keywords*-packet marking; IPv6 traceback; IPv6 extension header; network security.packet marking; IPv6 traceback; IPv6 extension header; network security.**

I. Introduction

One of the serious security shortfalls of the Internet today is failure to correctly identify the point of origin and the path taken by packets. The IPv6 source address of the packet can be easily forged by miscreants. The technique of forging source addresses is known as IP spoofing. The methods for fighting IP spoofing can be divided into two large areas — prevention and traceback [1].

Prevention methods seek to filter spoofed packets before they reach their destination and therefore seek to prevent the attack or minimize its effects. The methods proposed by Lee et al. [2], Liu et al. [3], and Shue et al. [1] are amongst the prevention methods.

Prevention methods have three serious disadvantages. Firstly, they may not recognize legitimate packets, and may therefore discard them [1]. Secondly, they require cooperation between different autonomous systems (ASes) on the Internet, requiring information exchange between them, and consequently requiring them to incur extra costs with storage and bandwidth resources. Thirdly, in filtering spoofed packets, they prevent not only the detection of the real perpetrator of the attack but also the obtainment of means of proving that he/she actually committed the attack. Thus, perpetrators continue to launch successive attacks on the Internet, unduly consuming its resources without being identified, held responsible and potentially penalized.

Traceback methods seek to identify the source of spoofed packets and therefore seek to identify the origin (or origins) of the attack. The best-known methods are probabilistic packet marking (PPM) and deterministic packet marking (DPM). The methods proposed by Savage et al. [4], and Goodrich [5] are amongst the PPM methods. The methods proposed by Belenky and Ansari [6], Xiang et al. [7], and Sun et al. [8] are amongst the DPM methods.

PPM methods proposed in the literature have some disadvantages. Firstly, they require large amounts of packets for correct reconstruction of the path between the origin and the destination of traffic. Secondly, they require the destination (or victim) to have significant computational resources in order to correctly identify the path taken by the packets. Thirdly, they fail to define protocols for the reliable exchange of keys between routers when making use of encryption or of hash functions. Finally, PPM methods cannot trace emails with spam content or distributed denial of service (DDoS) attacks.

DPM methods proposed in the literature also have some disadvantages. Firstly, they require the recipient (or victim) to know IP addresses of interfaces of border routers of the ASes to identify the source of the packets. Secondly, they require the recipient (or victim) to know which ASes deploy DPM and which do not. Thirdly, they do not specify any policy for the Internet service providers (ISPs) to deploy DPM either gradually or not. Finally, DPM methods do not propose any mark authentication scheme in order to guarantee the validity of the marks.

This paper proposes a new deterministic traceback method — chain-of-trust packet marking (CTPM). CTPM establishes a chain of trust composed of the border routers of the autonomous systems (ASes). It makes use of a new IPv6 extension header — the traceback extension header (TEH).

The paper is divided into six sections. Sections II and III describes the TEH and CTPM, respectively. Section IV details the future implementations through which CTPM will be evaluated. Section V presents the benefits of CTPM and Section VI concludes the paper.

II. Description of TEH

CTPM modifies neither the IPv6 header nor any of the existing IPv6 extension headers. CTPM just adds the new TEH to the IPv6 datagram. If the hop-by-hop options extension header is present in the datagram, the TEH must immediately follow it. Otherwise, the TEH must immediately follow the IPv6 header.

The creation of the new TEH header is necessary as none of the six existing IPv6 extension headers are currently processed exclusively by the border routers. The TEH is composed of the

 ISBN: 978-1-61208-450-3

two usual fields of any IPv6 extension header — next header (NH) and header length (HL) — [9], a padding field and the chain-of-trust mark (CTM). It is presented in Figure 1a.

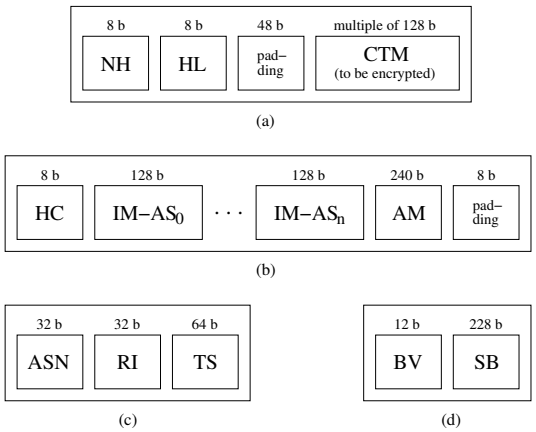


Figure 1. (a) traceback extension header (TEH); (b) chain-of-trust mark (CTM); (c) identification mark (IM); (d) autentication mark (AM) — b: bit(s)

CTM is composed of the hop count (HC) field, one or more identification marks (IM), an authentication mark (AM), and a padding field. It is presented in Figure 1b.

The HC indicates how many ASes the packet has passed through, and consequently, indicates how many IMs exist in the TEH. The IM is composed of three fields — autonomous system number (ASN), router interface (RI), and timestamp (TS). It is presented in Figure 1c.

The purpose of the ASN field is to globally identify an AS [10] through which the packet has passed. The RI field indicates the interface of the router through which the packet entered the AS. The 32 bits of the field can be used in the most convenient way for each AS. The TS field indicates the date on which the packet entered the AS. The date is represented in terms of Unix/Posix time.

The AM is composed of two fields — bit values (BV) and selected bits (SB). It is presented in Figure 1d. The purpose of the BV field is to store the value of 12 bits, selected randomly, of the datagram. The SB field identifies the position, in the datagram, of each selected bit.

Mark spoofing reduces the effectiveness of packet marking. Therefore, the use of encryption is necessary [2]. The operations community resists the use of any type of encryption on the Internet, mainly because of the costs involved [11]. However, the cost of encryption in this proposal can be largely offset by the cutting of costs with network appliances, caused by the reduction of garbage that currently circulates on the Internet.

The CTM size is always a multiple value of 128 bits. It is encrypted by the advanced encryption standard (AES) algorithm using a 128-bit key. In the vast majority of cases, the AS path does not exceed 8 hops [12]. Thus, the TEH can usually reach the maximum size of 168 (1 + 1 + 6 + 32 + 8 * 16) bytes. This size represents 13% of 1280 bytes — maximum transmission unit (MTU) garanteed by the network infrastructure [13]. Therefore, upon its creation, the IPv6 datagram must have a maximum size of 1112 bytes in order to circulate on the Internet without path MTU discovery [14].

III. Description of the CTPM method

In CTPM, marks are created and manipulated by border routers of the ASes which the packet traverses. They are stored in the TEH. Thus, each AS which a packet traverses can identify the path taken by the packet to reach it.

The AES key is shared only between the interface of the border router of an AS with its respective peer interface of the border router of the neighbouring AS. The keys could be established and exchanged by the two border routers through border gateway protocol (BGP) update messages. However, for safety reasons and to keep CTPM independent from BGP, CTPM will use Diffie-Hellman key exchange to establish and exchange AES keys. It will also use certificates of a public-key infrastructure (PKI), such as PKIX [15], to validate public keys. The AES keys must be changed periodically (every 60 days, for example).

Figure 2 illustrates the CTPM method by means of an example. In the example, there are 3 ASes — AS_0, AS_1 and AS_2 — each with two border routers. A packet is sent from the source computer to the destination (or victim) computer.

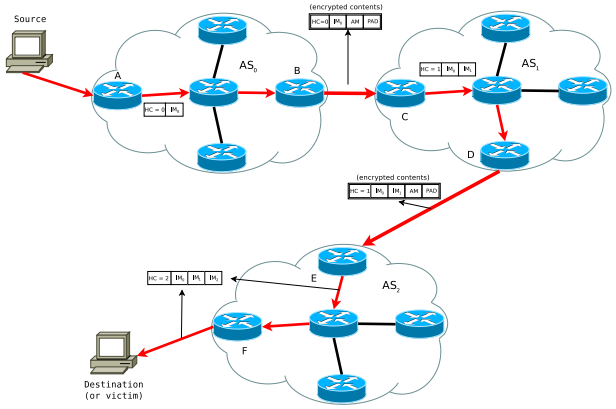


Figure 2. The chain-of-trust packet marking (CTPM) method

Upon receiving the packet sent by the source computer (administered by AS_0), border router A creates the TEH with HC value equal to 0 and with an IM of AS_0. The packet travels inside AS_0. Finally, upon receiving the packet, border router B adds the AM and padding fields to CTM. The CTM is then encrypted and TEH is rebuilt.

Upon receiving the packet sent by border router B, border router C decrypts the CTM and checks the AM, HC, and ASN fields. If the values of the three fields are valid, the packet processing continues. The AM and padding fields are then removed, the HC value is incremented, the IM of AS_1 is added to CTM, and TEH is rebuilt. The packet travels within AS_1. Finally, upon receiving the packet, border router D adds the AM and padding fields to CTM. The CTM is then encrypted and TEH is rebuilt.

 ISBN: 978-1-61208-450-3

Upon receiving the packet sent by border router D, border router E decrypts the CTM and checks the AM, HC, and ASN fields. If the values of the three fields are valid, the packet processing continues. The AM and padding fields are then removed, the HC value is incremented, the IM of AS_2 is added to CTM, and TEH is rebuilt. The packet travels within AS_2. Finally, upon receiving the packet, border router F removes the TEH from the packet, and sends the latter to the destination (or victim) computer. If required, border router F may save the packet header and TEH for offline analysis.

It is important to emphasize the fact that border router B trusts router A as they are both administered by AS_0. In turn, border router C trusts router B as both AS_0 and AS_1 know through which interfaces B and C are connected. They also know the AES key that they use to encrypt and decrypt the CTM fields of the TEH of the packets. In addition, both ASes exchange the AES key in a reliable manner via a PKI certificate. Successively, border router D trusts router C, router E trusts router D, and router F trusts router E. Thus, the CTPM method builds and makes use of a chain of trust in which its chain rings are border routers.

As may be seen from Figure 2, the gateway of the source computer is administered by AS_0. So, AS_0 is responsible for the traffic of the computer and it is up to AS_0 to take measures to avoid propagating malicious traffic coming from the computer. CTPM is intended to be incrementally deployed from the major carriers down to the ASes. Disincentives can also be applied, after deployment deadlines, by deployers for non-deployers as a mechanism to drive deployment.

IV. Future performance evaluation

Two CTPM implementations will be developed. The first will implement a Linux kernel module to extend the TCP/IPv6 stack. In the second, the CTM field will be encrypted and decrypted in field-programmable gate array (FPGA) hardware. The extra computational load on the border routers and extra network latency generated by CTPM will be measured and evaluated. These evaluations aim to verify the computational feasibility of CTPM and to give the ASes an idea of the extra investment required in network appliances if the CTPM method is adopted on the Internet.

V. Benefits of CTPM

The adoption of the CTPM produces several benefits for Internet security. The main benefit is the possibility of knowing the precise origin of each packet that circulates on the Internet. The sources of spam emails, which produce more than 70% of the global traffic of messages [16], can thus be identified and eliminated. Moreover, sources of attacks and malware can also be identified and fought.

Another benefit arises from the possibility of knowing the precise path, from origin to destination, of each packet that circulates on the Internet. CTPM can thus assist in both the correction of misconfigurations of BGP in order to ensure that each AS control- and data-plane paths match [17], and in the identification of bad actors in the routing system [18].

VI. Conclusions and future work

This paper proposes the new chain-of-trust packet marking (CTPM) method. CTPM establishes a chain of trust composed of the border routers of the autonomous systems (ASes). It makes use of the new IPv6 traceback extension header (TEH). The TEH contains encrypted marks that trace the path taken by each packet from its origin to the destination. Although the marks are created and manipulated only by the border routers, they do not allow the identification of internal topologies of the networks of the ASes traversed by the packet. The extra computational costs and network latency generated by CTPM will be measured and evaluated. These evaluations aim to verify the computational feasibility of CTPM and to give the ASes an idea of the extra investment required in network appliances if the CTPM method is adopted on the Internet.

Acknowledgment

This research is supported by CNPq and CAPES, Brazil.

References

[1] C. A. Shue, M. Gupta, and M. P. Davy, “Packet forwarding with source verification,” Computer Networks, vol. 52, 2008, pp. 1567–1582.

[2] H. Lee, M. Kwon, G. Hasker, and A. Perrig, “BASE: An incrementally deployable mechanism for viable IP spoofing prevention,” in Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), 2007, pp. 20–31.

[3] X. Liu, A. Li, X. Yang, and D. Wetherall, “Passport: secure and adoptable source authentication,” in Proceedings of the USENIX Symposium on Networked Systems Design (NSDI), 2008, pp. 365–378.

[4] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Network support for IP traceback,” Transactions on Networking, vol. 9, no. 3, 2001, pp. 226–237.

[5] M. T. Goodrich, “Probabilistic packet marking for large-scale IP traceback,” Transactions on Networking, vol. 16, no. 1, 2008, pp. 15–24.

[6] A. Belenky and N. Ansari, “On deterministic packet marking,” Computer Networks, vol. 51, 2007, pp. 2677–2700.

[7] Y. Xiang, W. Zhou, and M. Guo, “Flexible deterministic packet marking: an IP traceback system to find the real source of attacks,” Transactions on Parallel and Distributed Systems, vol. 20, no. 4, 2009, pp. 567–580.

[8] Y. Sun, C. Zhang, S. Meng, and K. Lu, “Modified deterministic packet marking for DDoS attack traceback in IPv6 network,” in Proceedings of the IEEE International Conference on Computer and Information Technology, 2011, pp. 245–248.

[9] RFC 6564, Internet Engineering Task Force, 2012, available at http://tools.ietf.org/html/rfc6564 [retrieved: January, 2016].

[10] RFC 6793, Internet Engineering Task Force, 2012, available at http://tools.ietf.org/html/rfc6793 [retrieved: January, 2016].

[11] K. Butler, T. R. Farley, P. McDaniel, and J. Rexford, “A survey of BGP security issues and solutions,” Proceedings of the IEEE, vol. 98, no. 1, 2010, pp. 100–122.

[12] Z. Gao and N. Ansari, “A practical and robust inter-domain marking scheme for IP traceback,” Computer Networks, vol. 51, 2007, pp. 732–750.

[13] RFC 2460, Internet Engineering Task Force, 1998, available at http://tools.ietf.org/html/rfc2460 [retrieved: January, 2016].

[14] RFC 1981, Internet Engineering Task Force, 1996, available at http://tools.ietf.org/html/rfc1981 [retrieved: January, 2016].

[15] Public-key infrastructure based on the X.509 protocol, Internet Engineering Task Force, available at http://datatracker.ietf.org/wg/pkix/documents/ [retrieved: January, 2016].

[16] Spam, Kaspersky Lab, available at http://www.kaspersky.com/about/news/spam [retrieved: January, 2016].

[17] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright, “Rationality and traffic attraction: incentives for honest path announcements in BGP,” in Proceedings of the ACM SIGCOMM Conference on Data Communication, 2008, pp. 267–278.

[18] G. Huston, M. Rossi, and G. Armitage, “Securing BGP — A literature survey,” Comm. Surveys & Tutorials, vol. 13, no. 2, 2011, pp. 199–222.

 ISBN: 978-1-61208-450-3

Towards Privacy in Identity Management Dynamic Federations

Lucas Marcus Bodnar*, Carla Merkle Westphall†, Jorge Werner‡ and Carlos Becker Westphall§
Department of Informatics and Statistics, Network and Management Laboratory
Federal University of Santa Catarina - UFSC
88040-970, Florianópolis, SC, Brazil
Email: {*lucas, †carla, ‡jorge, §westphall }@lrg.ufsc.br

***Abstract*—Dynamic federations allow users to access new service providers on demand. This dynamic access adds risks to personally identifiable information (PII) of users, since there are untrusted service providers. The federated identity management is essential to preserve privacy of users while performing authentication and access control in dynamic federations. This paper discusses characteristics to improve privacy in the dissemination of sensitive data of users in dynamic federations, proposing privacy scopes to be agreed in dynamic associations (federation time) among service providers and identity providers. A prototype of the dynamic federation and scopes agreement was developed using OpenID Connect.**

Keywords–Privacy; Dynamic Federation; OpenID Connect.

I. INTRODUCTION

Privacy refers to the ability of the individuals to protect information about themselves. Around the world many laws are being proposed in order to take care of privacy in the digital environment [1]. When privacy is considered, the actual attributes and data used for identification and access, have to be released with the user consent [2].

Federated Identity Management (FIM) enables the use of identities across organizations and they integrate control of identity data besides seeking to secure users through security mechanisms. Some federated identity management systems used today are Shibboleth and OpenID Connect [3][4].

The concept of dynamic federation is arising. It means a dynamic association between Identity Providers (IdP) and Service Providers (SP), at the time of access, without previously knowing each other and without the need for acceptance of previous agreements or contracts [5][6]. This flexibility in the federation framework should consider aspects of a trust model [7]. With dynamic federation scenarios risks with the privacy increase, since there is no previous trust establishment. The lack of trust among involved entities is a problem, because an entity becomes a member of the federation in a dynamic way, without a previous contract agreement [5][6].

The works [5], [6] and [7] discuss dynamic federation architectures and they all use the Security Assertion Markup Language (SAML) standard to set up the federation. SAML does not allow creating a dynamic federation and although some works propose dynamic federation with SAML, most of them require changes in the specification flows [5]. Moreover, the majority of these works that propose dynamic federations do not address privacy [5][7].

The main goal of this work is to improve privacy in dynamic federation environments, proposing a solution for privacy scopes agreement in the dynamic association between SP and IdP, improving the user's privacy. This work uses the OpenID Connect to build the federation. The OpenID Connect (OIDC) uses JavaScript Object Notation (JSON) Web Tokens instead of SAML and unlike the SAML based federations, the OpenID Connect has a good support for dynamic associations.

This paper is organized as follows: Section II provides a background of the concepts; related works are described in Section III; Section IV presents the federation concept in OpenID Connect; the proposal of privacy scopes is presented in Section V and the initial results are described in Section VI. Finally, conclusions and future work are presented in Section VII.

II. BACKGROUND

This section presents a brief overview on privacy and identity management.

A. Privacy

Privacy is to choose when, how and to what extend personal information can be released to others. A way to achieve privacy is using the *Privacy-Enhancing Technologies* (PET) [8] that aim at protecting the individual's privacy by providing anonymity, pseudonymity, unlinkability and unobservability to the users. Anonymity is when the user cannot be identified in a set of users. Pseudonymity is when pseudonyms are used to identify the user instead of the real identifier. Both anonymity and pseudonymity are desirable principles and the user should be able to choose according to his preferences [2].

B. Basic Identity Management Concepts

Identity management can be defined as the process of creation, management and use of identities and the infrastructure that provides support for this set of processes [4][9].

Bertino and Takahashi, in [9], presented the roles that exist in an identity management system:

- Users – entities that want to access some kind of service or resource;
- Identity – set of attributes that can be used to represent a user, also called Personally Identifiable Information (PII);
- Identity provider (IdP) – has the responsibility to manage users PIIs, perform the authentication process and disseminate data to SPs;
- Service provider (SP) – delivers the resource/service desired by a user. It delegates the process of authentication to IdPs and usually is responsible for the authorization process that is performed using the disseminated set of attributes from the IdP.

Chadwick [10] defines that a federation is an association of SPs and IdPs.

There are tools that can be used to create federated environments; some use SAML to exchange data between IdPs and

 ISBN: 978-1-61208-450-3

SPs such as Shibboleth [11]; while others use JSON such as OpenID Connect protocol [12]. The OpenID Connect [12] was chosen to serve as the basis for our implementations because it is open source, it has a standard protocol, has a native support for dynamic associations and as it uses a lightweight message format (JSON), it fits for mobile environments.

Figure 1 presents how the OIDC workflow when unauthenticated users request a protected resource.

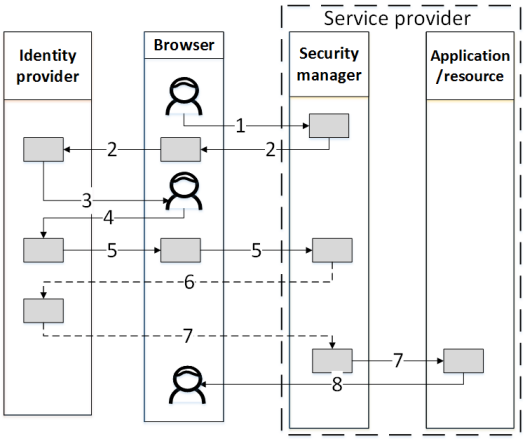


Figure 1. OpenID Connect work flow [13]

In Figure 1, the OpenID Connect interaction flow among user (browser), IdP and SP is shown. In step 1, the user requests access to the protected resource. In step 2, the SP requests user authentication which is performed in step 3. In step 4, the user can get a chance to consent to the release of data. In step 5, the IdP sends proof of authentication to the SP, that is validated in step 6. In step 7, the SP obtains user attributes from the IdP to release or not access to the resource in step 8.

C. Identity Management Concepts in OpenID Connect

The terminology used to represent the identity management concepts and technologies in OIDC is described in this section [3][14].

OpenID Provider (OP) – is the Authorization Server of OAuth 2.0 and is able to authenticate the final user and provide Claims to a relying party about the authentication and the final user. The OP is the IdP (*Identity Provider*) according to the traditional concept;

Relying Party (RP) – is the Client application of OAuth 2.0 that requires end-user authentication and Claims from OP. The RP is the SP (*Service Provider*) according to the traditional concept;

Claim – information about an entity, such as name and phone number;

JSON – is a lightweight, text-based and language-independent data exchange format;

JSON Web Token (JWT) – is a compact, URL-safe manner to represents claims to be transferred between two parties. A JWT is a string that represents a set of claims as a JSON object that is encoded using JWS (*JSON Web Signature*) or JWE (*JSON Web Encryption*), allowing claims to be digitally signed, MACed or ciphered;

Pairwise Pseudonymous Identifier (PPID) – Identifies the entity to a Relying Party. It cannot be correlated with the entity's PPID at another Relying Party.

III. Related works

The work described in [6] proposes an architecture according to the SAMLv2/ID-FF standards. The architecture considers federated identity management, the assessment of reputation and customization of the privacy preferences of user data. There is a lack of detail about interactions among parties. The work uses a reputation metric for trust establishment. In the case of privacy preferences, it only describes some possibilities for exchanging data, such as using IdP policies.

The work presented in [7] discusses a dynamic architecture to establish identity federations, based on a reputation exchange protocol. The protocol to collect data reputation is decentralized, storing the reputation of the providers.

The paper [15], which is a work developed in our security research group, presents an approach to address the issues involving privacy in the identity management systems. The proposal addresses three issues: the lack of PII control of users, the lack of models to assist users in data dissemination during the interaction and, the lack of user preferences guarantees on the SP side.

The work [16] identifies the privacy features relevant to authentication technologies and determines what privacy features are provided by each authentication technology. Some authentication technologies compared are OpenID Connect, OAuth, Shibboleth, Idemix pseudonym and U-Prove token.

The research described in [5] addresses the management of dynamic identity federations. The proposal allows users to create federations dynamically between two prior unknown organizations. The issue of trust is also a key component of the discussions. Trust means the way IdP and SP federate: *fully trusted* entities have a legal contract between IdP and SP; *semi-trusted* entities are the SPs that have been added dynamically to an IdP inside a federation by a user but without any contract between IdP and SP; *untrusted* entities are the IdP and SP added dynamically without the presence of any contract. A proof of concept is discussed using SAML federations and use-cases demonstrate the ideas proposed.

IV. OpenID Connect Federation

Federated identity management, according to [3], is a setup where identity is shared across domains.

An identity federation is a trust relationship between a service provider and identity provider, i.e., the service provider trusts the identity provider to authenticate the user and to provide user attributes (if necessary). The user tries to access a service provider located in different domains and is redirected to his own identity provider where he is authenticated and the result of this authentication process is sent to the service provider [17].

Third-party authentication is named federated authentication when a third-party, such as an identity provider and the service provider, belong to different organizations [16].

An OIDC Federation is an association among relying parties (RP) and OpenID Providers (OP). Even not considered a federation [18], a simple association between a RP and an OP of a single domain is also possible in OIDC.

 ISBN: 978-1-61208-450-3

Although in other federation platforms, trust is achieved through the acceptance of contracts, manual key or metadata exchanges in an explicit way [19], in OpenID Connect this association occurs in a more simplified way, through the registration of RP in OP [14][18].

This RP registration in OPs, such as Google or MitreID, is performed through a registration page and, in this moment, terms of use are accepted and combined. The authentication process of the own RP is also defined to give the RP the ability to receive user information from OP. So, an entity responsible for the RP provides the redirecting Uniform Resource Identifiers (URIs) and the other information required by the OP, such as the application name, website and description, for example. After supplying the information required, the RP receives from the OP a client identification (ID) and a password (ClientSecret) that will be used to authenticate the RP. The authentication of the RP can be executed in three ways [14][20]:

- In a default way, using the ID and the ClientSecret to the authentication via HTTP Basic authentication scheme;
- Including the ID and the ClientSecret in the request body;
- Using the ClientSecret as a shared key for creating a JWT using an HMAC SHA algorithm containing some required claim values.

At registration there is also the possibility of registering a public key to be used in the authentication, rather than the ClientSecret methods, i.e., the RP signs a JWT containing some required claim values with this key.

After the registration process, a trust relationship is established between the RP and the OP. In this work, we consider that when an OP allows an RP to register itself, a *federation* (or trust) can be established in OIDC.

In this way, an OP in a domain can federate with other RPs in other domains. So, a user can authenticate in his own domain and use RPs in other domains.

V. Proposal of Privacy in Identity Management Dynamic Federations

In federation systems, the identity providers are used for user authentication, to store collections of user's attributes and to manage these identities (in some systems, the attributes are managed by a separate part, called attribute provider) [4]. Generally, after the user authentication, the user gives the consent to the release of his identity and attributes to RP and thereafter, the OP sends these data to RP.

There is a lack of trust in dynamic associations since it is possible to associate any RP to an OP. It is necessary to control the sending of identities and attributes information to these RPs. Thus, to improve the privacy of dynamic environments, we propose a better control of users over his data, using privacy scopes that support anonymous access, use of pseudonyms and a scope that releases only partial attributes to RPs.

Once these privacy scopes are supported, it is still necessary to perform the combination and agreement of the scopes in a dynamic way, at the time of RPs and OPs association, to allow the use of services on demand by users with enhanced privacy.

Despite Shibboleth's support for pseudonyms and anonymity, it has a lack of support for dynamic federations [4][5]. OIDC does not support privacy scopes, but has a great support for dynamic associations.

A. Dynamic Federation in OpenID Connect

There is also other way of registering RPs on OPs in OIDC by specifically using the dynamic protocol known as *DynamicClientRegistration* [20]. It is possible to register dynamically an RP by sending RP's metadata to the OP. Despite using metadata as in other federation platforms, client metadata is used more dynamically, since the OP can change the information on it and is only used the RP's metadata, it is not necessary and indispensable (such as in Shibboleth/SAML federations) for an RP to have OP metadata.

Figure 2 shows the dynamic registering of RPs in OIDC.

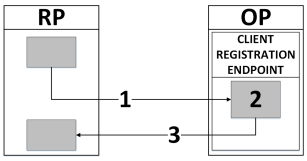


Figure 2. Dynamic Client Registration Flow

1. RP registration request - the RP sends an HTTP post message, with the content type application/json, to the client registration endpoint with parameters of client metadata. The RP chooses these registration parameters among the list of OP supported values. In OIDC standard only target/redirect URIs are required, other metadata are optional. In Figure 3, a post example is presented, with header and metadata;

2. RP registration - the OP assigns to this RP a unique client identifier, assigns a client secret or registers the public key of the RP and associates metadata sent in the request to the client identifier issued. The OP can still provide default values for any missing item in client metadata, reject or change any values to acceptable values.

3. RP registration response - with the success of the registration, the OP returns the code HTTP 201 Created, a JSON document (type of content is application/json) and all metadata registered for this RP, including the fields modified by the OP plus the RP identifier and if applicable, the client secret.

```
1 POST /register HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 Host: server.example.com
5
6 {
7   "redirect_uris": [
8     "https://client.example.org/callback",
9     "https://client.example.org/callback2"],
10  "client_name": "My Example Client",
11  "client_name#ja-Jpan-JP":
12     "\u30AF\u30E9\u30A4\u30A2\u30F3\u30C8\u540D",
13  "token_endpoint_auth_method": "client_secret_basic",
14  "logo_uri": "https://client.example.org/logo.png",
15  "jwks_uri": "https://client.example.org/my_public_keys.jwks",
16  "example_extension_parameter": "example_value"
17 }
```

Figure 3. Post of RP Registration

 ISBN: 978-1-61208-450-3

The metadata file is composed of information about RPs. This metadata file, in the static registration, is manually generated with the help of the registration page, but, in the dynamic way, it is sent within the HTTP post content. This client metadata file has mandatory parameters such as redirecting URIs or optional parameters as the client name, form of authentication and used algorithms [20].

Figure 3 presents an example of RP metadata: lines 1 to 4 are the header and lines from 6 to 17 describe metadata chosen by the own RP in the registration moment, of RP in OP.

All this OpenID Connect registration can be performed dynamically, without the need of previous agreements, without OPs needing to know RPs and even then, it is possible to establish this association at the time of access among different domains. We believe this process represents the concept of *dynamic federation*.

There are few references in the literature about dynamic federation. In [5], it has the following definition: "a dynamic federation with respect to the identity management is a business model in which a group of two or more previously unknown parties federate together dynamically without any prior business and technical contract with the aim to allow users to access services under certain conditions".

We think this concept is covered in OIDC through the *Discovery Protocol*, used to discover information about the OP, and the *DynamicClientRegistration Protocol*, which enables the RP to dynamically register with the OP. Thus, this work considers a *dynamic federation establishment* when the RP registration with the OP is done dynamically, across different domains.

As mentioned before, this dynamic registration can also be used in associations in the same domain and our study will work in this case too (even not being recognized as a federation and not being our focus to consider the same domain).

The great advantage of a dynamic association is to be able to associate the OP and RP in real time, without the need for prior agreements, and so, some problems can arise, mostly related to trust. OpenID Connect with dynamic registration considers the OP and RP as trusted and its use is a user's choice, since the OP and RP are able to match the settings required for proper communication between them.

B. Privacy in OpenID Connect

The user's PIIs are stored in the OP, and to ensure privacy, OIDC asks for user confirmation to release data to RP. After user confirmation, RP requests user data (PIIs) directly from the OP and obtains user identity and user's attributes.

To ensure better privacy in OIDC, some privacy scopes to access the RP are proposed in [15]. These scopes are agreed upon previously, between RP and OP, and among those supported scopes, the end user chooses the one that best fits with his privacy preferences. These scopes are:

Access with Total Attributes: Sending total attributes is the standard of OIDC. When the RP makes a request of the attributes and the user identification, the user allows or not the OP to send this information;

Access with Partial Attributes: The user chooses the attributes that he wants to send to the RP. The OP shows to the user the attributes list and the user selects the attributes to be released, helping users to assess and determine the data they want to send to the RP. In addition, the user can encrypt the selected attributes with his public key, in order to control the spread of data [13][21];

Access with Pseudonym: The RP receives the authentication data, the pseudonym of the user and also can receive some data that cannot be linked to user identification. The use of pseudonyms maintains the privacy of the users, and in the RP, the user is identified by his nickname. The identification of users is only permitted in the original OP, in case of legal request, for auditing purposes. The nickname must be issued dynamically through use of random numbers linked to the user identifier. Thus, attackers must have difficulty in correlating historical data with the user's identity. The OIDC recommends the use of a *Pairwise Pseudonymous Identifier* (PPID) to protect the user from a possible correlation among RPs [14]. To implement access with pseudonyms, in this work, before sending the ID Token to the RP, the OP maps the subject identifier of the user (named sub claim), with a random nickname for each RP. To implement pseudonyms, a scope claim is also added, using the parameter *pseudonym_profile*. Besides, the RP can only request user data that cannot be linked to user identification;

Access with Anonymity: The user accesses the RP without any type of identification that can compromise his privacy. The user can access the resource after the authentication on OP. The RP receives the authentication data but without user identifier or PIIs. The RP receives the authentication data, with information about the authentication, like identification of RP that the authentication was issued, identification of the OP that issued the authentication, authentication expiration time and other data, but without user identifier or PIIs. With that, the system avoids the identity correlation attacks or linkability of users' data. There are many projects used to achieve anonymity [4], like U-Prove, Idemix and P-IMS. In this work, we use the OIDC flow, adding the scope claim of type *anonym_profile* and without the sub claim, PIIs or identification data of the users. So, without any information about the user, we achieve the concept of anonymity. Thus, there is no reference of user identification, only a proof of authentication and non-linkable data, ensuring anonymity of the user in the RP. This kind of anonymity is interesting in cases that, to access certain services, the user has to prove that he belongs to a certain domain. For example, in university federations, the user has to prove that he belongs to the federation to access a service.

VI. Dynamic Agreement of Scopes and Initial Results

A prototype of the proposed solution was developed based on an identity management system running on a cloud environment. The identity management system chosen was the OpenID Connect, since it is a highly detailed framework, documented and extensible. We used the code developed by the specification under the name MITREid Connect, available in Java by MIT (Massachusetts Institute of Technology) [22].

In this work, the aim is to find a way to allow a combination of additional scopes (anonym, pseudonym and partial attributes) in the dynamic registration. Thus, a better privacy in dynamic federations is achieved.

Previously, the implementation of scopes in each OP and RP was necessary, to support these additional scopes. So, we

 ISBN: 978-1-61208-450-3

propose that in the registration of RP on OP, the combination of scopes have to be performed. Thus, at the moment of registration, using the DynamicClientRegistration protocol, the scopes supported by the RP are sent along with the Client Metadata, as shown in Figure 4.

```
1 POST /register HTTP/1.1
2 Content-Type: application/json
3 Accept: application/json
4 Host: server.example.com
5
6 {
7   "redirect_uris": [
8     "https://client.example.org/callback",
9     "https://client.example.org/callback2"],
10  "client_name": "My Example Client",
11  "client_name#ja-Jpan-JP":
12     "\u30AF\u30E9\u30A4\u30A2\u30F3\u30C8\u540D",
13  "token_endpoint_auth_method": "client_secret_basic",
14  "logo_uri": "https://client.example.org/logo.png",
15  "jwks_uri": "https://client.example.org/my_public_keys.jwks",
16  "example_extension_parameter": "example_value"
17  "scopes_supported": [
18    "partial_attribute_profile"
19    "pseudonym_profile"
20    "anonym_profile"]
21 }
```

Figure 4. Client Metadata example with Privacy Scopes

In Figure 4, lines 17 to 20 are added with the scopes supported by the RP and this metadata will be sent to the OP, to make the registration of the RP.

Thus, OP will check the scopes supported by RP and itself, and will register the RP with the scopes supported by both. For example, if the RP supports the scopes of partial attributes, pseudonym and anonym, it will send the scopes via metadata to the OP and, if the OP only supports partial attributes and pseudonym scopes, it will return the metadata only with the partial attributes and pseudonym scopes, which is supported by both. If the OP return no scopes, the RP will know that the OP does no support privacy scopes, and so, the default scope of OIDC will be used.

After the scopes agreement during RP dynamic registration on OP, it is necessary for the user to choose the scopes in order to use a service, performing the following steps:

1) On a user's access, the RP shows a WAYF (Where Are You From) page, where the user's OP is inserted.
2) The RP will verify if the OP is already registered. If already registered, the RP will go to step 3. If not, it will register the OP, agreeing to the privacy scopes supported by both through DynamicClientRegistration protocol.
3) The RP requests user authentication, it redirects the user to the OP to the authentication process.
4) The OP asks the user about his credential to perform the authentication process.
5) After the authentication process, the OP asks the user what is the desired privacy scope (total, partial, pseudonym or anonym), based on the privacy scopes supported by both RP and OP, as shown in Figure 5. The OP also informs the user about the chosen scope to access the RP, the attributes that will be sent to RP, the reason of the data dissemination and obtains the permission to release these attributes to the RP.
6) The user (browser) is redirected to the RP with a ticket to confirm the authentication process in the OP.

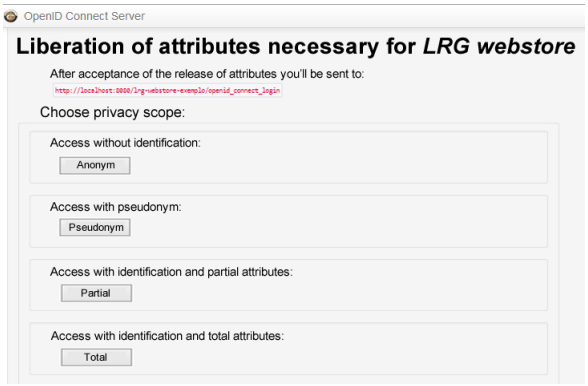


Figure 5. Privacy scope selection in OP

7) The RP intercepts the request, validates the proof of authentication and obtains an access token to gather some extra data about the user. In the case of scopes which do not allow sending PIIs: a) it is possible that no user attributes are obtained; b) only attributes that cannot be linked to user's identification are obtained.
8) The RP calls the OP with the access token to get attributes required to perform the RP's desired service, but again, only using attributes allowed by the user according to the privacy scope used.
9) The desired service/resource is delivered.

There was no change in the standard OIDC flow. Extensions were developed in the execution of OIDC actions, performed in steps 2, 4 and 5. In step 2, there is a dynamic agreement of scopes; in step 4, the user chooses the desired privacy scope to access the service/resource; in step 5, the user consents to release attributes according to the chosen privacy scope. So, it is important that the OP knows which attributes can be sent to the RP depending on the scope and the user should be aware about the usage of his data. For example, it is not allowed to send user PIIs on pseudonymity or anonymity scopes.

VII. Conclusion and Future Works

Our contribution in this work was to improve privacy in dynamic federations, proposing privacy enhancing technologies that use the dynamic association support to federate in OpenID Connect. So, users have a better control over their identity and attributes. An extension of OIDC access profiles was proposed, to add different levels of privacy. Considering [4] and [16], we can conclude that these developed extensions improved properties of OpenID Connect such as pseudonymity, anonymity, unlinkability, undetectability and confidentiality. With our proposed extension we achieve pseudonymity, anonymity and unlinkability in OIDC, properties that did not exist in OIDC according to [16].

The problem of agreeing to these privacy scopes dynamically was solved along with the DynamicClientRegistration protocol, through which these scopes were combined using the RP metadata, without any changes in the dynamic registration flow of OpenID Connect.

Our work combines the concept of dynamic federation with privacy features proposed in [15] and how to agree to these concepts dynamically. With that, our work differs from [5] and

 ISBN: 978-1-61208-450-3

[7] by defining and applying theoretical and practical concepts of privacy in dynamic federations.

The paper of Sanchez et. al. [6] also works with privacy in dynamic federations, building a dynamic federation model with a privacy layer and a trust layer. In contrast, our work extends a protocol already implemented and used by a large number of organizations, the OpenID Connect, without modifications in its specifications.

The following related works present a metric (reputation or risk) to measure the trust of the RPs: [6][7][15]. Different levels of trust in dynamic federations are cited in [5].

Another difference compared to related works, is that we have developed a prototype implementation using OpenID Connect, that works with JSON instead of SAML, which makes it easier to use in mobile environments and offers native support to dynamic associations. So far, the work has not changed the flow of interaction between RP and OP. In this way, new threats will not happen because of our model.

However, a limitation of this work is that it has an initial prototype and it lacks tests and measurements of how the federation will behave. It also lacks a trust level model.

We plan as futures works: a) to implements levels of trust (e.g., considering metrics of risks and reputation); b) to test the scalability of the federation; c) to study other forms to ensure user privacy, especially after user data is released to the RP.

REFERENCES

[1] European Parliament and the Council of the European Union, "Directive 95/46/ec of the european parliament and of the council," [retrieved: January, 2016]. [Online]. Available: http://ec.europa.eu/justice/policies/privacy/docs/95-46-ce/dir1995-46_part1_en.pdf

[2] A. Pfitzmann and M. Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology. [retrieved: October, 2015]. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/ (2010)

[3] G. Alpár, J. henk Hoepman, and J. Siljee, "The identity crisis security, privacy and usability issues in identity management," 2011, [retrieved: November, 2015]. [Online]. Available: http://arxiv.org/abs/1101.0427

[4] E. Birrell and F. Schneider, "Federated identity management systems: A privacy-based characterization," Security Privacy, IEEE, vol. 11, no. 5, Sept 2013, pp. 36–48.

[5] S. Ferdous and R. Poet, "Managing Dynamic Identity Federations using Security Assertion Markup Language," Journal of theoretical and applied electronic commerce research, vol. 10, 05 2015, pp. 53 – 76.

[6] R. Sanchez, F. Almenares, P. Arias, D. Diaz-Sanchez, and A. Marin, "Enhancing privacy and dynamic federation in idm for consumer cloud computing," Consumer Electronics, IEEE Transactions on, vol. 58, no. 1, February 2012, pp. 95–103.

[7] P. Arias Cabarcos, F. Almenárez, F. Gómez Mármol, and A. Marín, "To federate or not to federate: A reputation-based mechanism to dynamize cooperation in identity management," Wirel. Pers. Commun., vol. 75, no. 3, Apr. 2014, pp. 1769–1786.

[8] Y. Wang and A. Kobsa, "Privacy-enhancing technologies," 2008, [retrieved: December, 2015]. [Online]. Available: http://www.cs.cmu.edu/afs/cs/Web/People/yangwan1/papers/2008-Handbook-LiabSec-AuthorCopy.pdf

[9] E. Bertino and K. Takahashi, Identity Management: Concepts, Technologies, and Systems. Norwood, MA, USA: Artech House, Inc., 2010.

[10] D. W. Chadwick, "Federated identity management," in Foundations of Security Analysis and Design V. Springer, 2009, pp. 96–120.

[11] Shibboleth, "What's shibboleth?" 2014, [retrieved: July, 2014]. [Online]. Available: https://shibboleth.net/about/

[12] OpenID, "Welcome to openid connect," 2014, [retrieved: March, 2015]. [Online]. Available: http://openid.net/connect/

[13] R. Weingärtner, "Dissemination control of sensitive data in federated environments," Master's thesis, UFSC, Brasil, 2014.

[14] N. Sakimura, J. Bradley, M. B. Jones, B. de Medeiros, and C. Mortimore, "Openid connect core 1.0," 2014, [retrieved: August, 2015]. [Online]. Available: http://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication

[15] J. Werner, C. M. Westphall, R. Weingartner, G. A. Geronimo, and C. B. Westphall, "An approach to idm with privacy in the cloud," in Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on, Oct 2015, pp. 168–175.

[16] F. Corella and K. Lewison, "Privacy postures of authentication technologies," in The Internet Identity Workshop, ser. IIW 2013, Mountain View, CA, 2013, [retrieved: September, 2015]. [Online]. Available: http://pomcor.com/techreports/PrivacyPostures.pdf

[17] D. Chadwick, K. Siu, C. Lee, Y. Fouillat, and D. Germonville, "Adding federated identity management to openstack," Journal of Grid Computing, vol. 12, no. 1, 2014, pp. 3–27.

[18] D. Hardt, "Rfc6749-the oauth 2.0 authorization framework-revision," 2012, [retrieved: October, 2015]. [Online]. Available: http://tools.ietf.org/html/rfc6749

[19] D. R. D. Santos, T. J. Nascimento, C. M. Westphall, M. A. P. Leandro, and C. B. Westphall, "Privacy-preserving identity federations in the cloud: A proof of concept," Int. J. Secur. Netw., vol. 9, no. 1, Feb. 2014, pp. 1–11.

[20] N. Sakimura, J. Bradley, M. B. Jones, B. de Medeiros, and C. Mortimore, "Openid connect dynamic client registration," 2014, [retrieved: September, 2015]. [Online]. Available: http://openid.net/specs/openid-connect-registration-1_0.html

[21] R. Weingärtner and C. M. Westphall, "Enhancing privacy on identity providers," in SECURWARE 2014, Portugal, Nov 2014, pp. 82–88.

[22] MIT, "Mitreid connect," 2014, [retrieved: October, 2014]. [Online]. Available: https://github.com/mitreid-connect/OpenID-Connect-Java-Spring-Server

 ISBN: 978-1-61208-450-3

A Packet-Interleaving Scheme for Reliability Resilience under Burst Errors in Wireless Sensor Networks

Tsang-Ling Sheu and Yen-Hsi Kuo
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
sheu@ee.nsysu.edu.tw ysk@atm.ee.nsysu.edu.tw

***Abstract* — This paper presents a packet interleaving scheme (PIS) for increasing packet reliability under burst errors in wireless sensor networks (WSN). The proposed PIS, using Reed-Solomon (RS) codes, classifies data into two types: high-reliability-required (HRR) data and non-HRR data. An HRR packet is encoded with a short RS symbol, while a non-HRR packet with a long RS symbol. When an HRR and a non-HRR packet arrive at a sensor, they are interleaved on a symbol by symbol basis. Thus, the effect of burst errors (BE) is dispersed and consequently the uncorrectable HRR packets can be reduced. For the purpose of evaluation, two models, the uniform bit-error model (UBEM) and the on-off bit-error model (OBEM), are built to analyze the packet uncorrectable probability. In the evaluation, we first change the lengths of BE, then we vary the shift positions in a BE period, and finally we increase the number of correctable symbols to observe the superiority of the proposed PIS in reducing packet uncorrectable probability.**

Keywords: WSN, RS Code, burst errors, interleaving, packet uncorrectable probability

I. INTRODUCTION

Along with the increasing requirements for quality of living and home security, sensors have been widely deployed inside or outside a building to collect environmental information, such as temperature, humidity, image, motion picture, etc. To effectively deliver the collected data back to a control center for further analysis, a wireless sensor network (WSN) [1-3] is usually built. However, packet transmission over a WSN may encounter intermittent errors due to weak signals or interferences. The erroneous packets, if comprising of text or numbers, such as temperature or humidity, would require packet re-transmission, which increases network load. Thus, the motivation of this paper is to increase transmission reliability over a WSN, which has recently attracted many researchers' attention.

Basically, previous researches on transmission reliability over a WSN can be divided into two major categories: reliable routing and information coding. In the first category, to increase the transmission reliability after data are collected by a sensor node, relay nodes (RNs) are employed. For examples, H. Chebbo, et al. [4] modified IEEE 802.15.4 MAC frames. The authors added one bit in the frame control field, with which whether it is necessary to build a tree by RNs or just build a simple star, can be determined. Moreover, R. Sampangi, et al. [5] utilized RNs to divide sensors into several cluster networks. Since the distance from a sensor to its cluster head is reduced, the quality of data transmission is greatly improved. To protect the routing path, S. Kim, et al. [6] utilized both coding and retransmission schemes once the established path fails. However, in these schemes, it is inevitable that end-to-end packet delay will increase accordingly due to multiple-hop forwarding.

Thus, in the second category, instead of developing reliable routing, the authors switch their interests to information coding. For examples, E. Byrne, et al. [7] designed a coding scheme which can increase the probability of successful decoding based on graph theory. Y. Hamada, et al. [8] proposed a scheme to reduce packet error rate by using Luby Transform (LT) codes [9]. Their proposed scheme has achieved small complexity of O(n); yet too many packets require retransmission when bit error rate is high. Thus, K. Ishibashi, et al. [10] proposed an embedded forward error control (FEC) technique which utilizes RS (Reed Solomon) code to reduce packet error rate. Similarly, M. Busse, et al. [11] can recover lost chunks by using Fountain code and Raptor code. To increase data reliability and processing speed, K. Yu, et al. [12] designed a new FEC which protects header and payload, respectively. Similarly, M. Srouji, et al. [13] proposed a reliable data transfer scheme which can adjust the lengths of redundancy code based on the successful receiving rate at the downstream node.

Unlike the previous research work, in this paper we propose a packet interleaving scheme (PIS) to reduce the impact of burst errors (BE) on high-reliability-required (HRR) data in WSNs. Although Reed-Solomon (RS) codes may correct bit errors under certain constraints, it may not be economically worthy in dealing with burst errors when the number of consecutive bit errors exceeds a threshold. Hence, to increase packet correctable probability in a WSN, the proposed PIS first classifies the collected data into two different types: HRR data and non-HRR data. An HRR packet is encoded with a short RS symbol, while a non-HRR packet with a long RS symbol. When an HRR and a non-HRR packet arrive at a sensor, they are interleaved on a symbol-by-symbol basis. The noticeable benefit from the packet interleaving is that the burst errors are dispersed and the uncorrectable probability of HRR packets is significantly reduced.

The remainder of this paper is organized as follows. In Section 2, the proposed PIS and its operations are described. In Section 3, an analytical model is built using two bit-error models, the uniform bit-error model (UBEM) and the on-off bit-error model (OBEM). In Section 4, numerical results are presented and discussed. Finally, conclusions are drawn in Section 5.

 ISBN: 978-1-61208-450-3

II. PACKET INTERLEAVING SCHEME

A. WSN with Multi-hop Tree Structure

In a wireless sensor network (WSN), a coordinator is the sink which gathers all the data collected from other distant sensor nodes. To facilitate data gathering from all the sensor nodes, it is very constructive that the coordinator and the sensor nodes will collaborate to build a multi-hop tree structure (MTS), as shown in Figure 1. In an MTS-based WSN, data collected by a sensor node will be forwarded hop-by-hop to the coordinator. Thus, by fully utilizing a branch node of the MTS, in this paper, we propose a packet interleaving scheme (PIS) based on RS codes to reduce the impact of burst errors on packet uncorrectable probability.

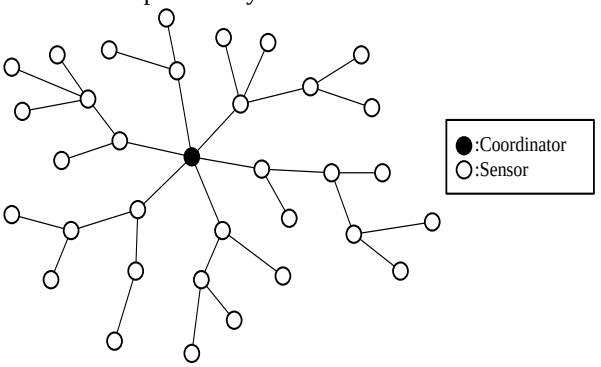


Figure 1. WSN with multi-hop tree structure

B. Packet Interleaving

In the proposed PIS, packets collected by a sensor node are classified into two different types: high-reliability required (HRR) packet and non-HRR packet. An HRR packet is defined as a packet which requires for retransmission, if uncorrectable burst errors exist. Payload in an HRR packet consists of numerical data, such as temperature, moisture, luminance, etc. This type of packet has relatively shorter data length (usually, a couple of bytes) and each uncorrectable HRR packet requires for retransmission. Hence, it is better to employ a shorter-length symbol (in this paper, we use m = 4) to encode an HRR packet with shorter data length. On the other hand, payload in a non-HRR packet consists of non-numerical data, such as video, audio, etc. This type of packet has relatively longer data length (usually, in the order of kilo bytes) and each uncorrectable non-HRR packet may not require retransmission. Thus, it is better to employ longer-length symbol (we use m = 8) to encode a non-HRR packet with longer data length.

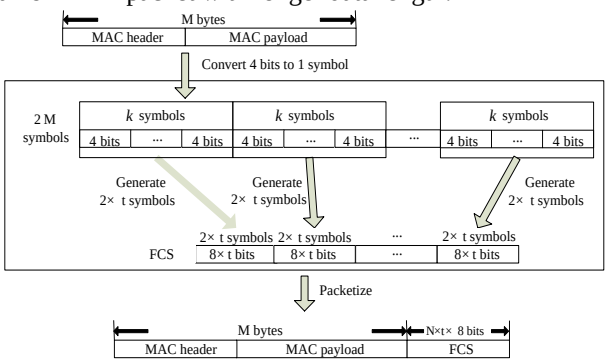


Figure 2. An HRR encoded with m = 4

As it is illustrated in Figure 2, an HRR packet is encoded with a shorter-length RS symbol (i.e., m = 4). First, an M-byte MAC-layer header and payload is converted to $\frac{(M \times 8)}{4} = 2M$ symbols on the basis of 4 bits per symbol. Thus, we have the length of a codeword is n, where $n = 2^m - 1 = 2^4 - 1 = 15$, the number of symbols for user data in a codeword is k, where $k = n - 2 \times t = 15 - 2 \times t$, and the number of symbols for redundancy code in a codeword is $2 \times t$. Let $N = \left\lceil \frac{2M}{k} \right\rceil$, which denotes the number of codeword required for encoding the M-byte packet (header plus payload). The total redundancy code (or FCS) is therefore equal to $N \times 2 \times t$ symbols, or $N \times 2 \times t \times 4 \,(= N \times t \times 8)$ bits.

Similarly, as it is illustrated in Figure 3, a non-HRR packet is encoded with a longer-length symbol (m = 8). An M-byte MAC-layer header and payload is converted to $\frac{(M \times 8)}{8} = M$ symbols on the basis of 8 bits per symbol.

A code-word length, $n = 2^m - 1 = 2^8 - 1 = 255$ bytes, is much greater than the maximum length of a packet (127 bytes in WSN). Thus, a code-word is sufficiently enough to encode a MAC-layer packet. Thus, the length of redundancy code (or FCS) is equal to $2 \times t$ bytes.

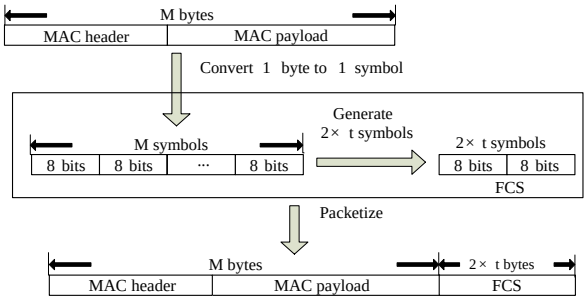


Figure 3. A non-HRR encoded with m = 8

When both an HRR and a non-HRR packet are received by a branch node in a tree-structured WSN, these two packets are interleaved on a symbol-by-symbol basis, as shown in Figure 4. The interleaved packet is then forwarded to an upper stream node, which performs decoding and correction process. However, as shown in Figure 5, an interleaved packet may not be correctable, if it encounters burst errors where the number of total errors is greater than t. In the proposed PIS, by separating the interleaved single packet back to their original two packets, each individual packet may become correctable. This is because the number of errors in each separated packet is highly possible to be smaller than t.

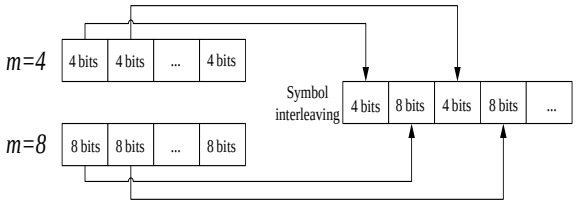


Figure 4. Packet interleaving on a symbol-by-symbol basis

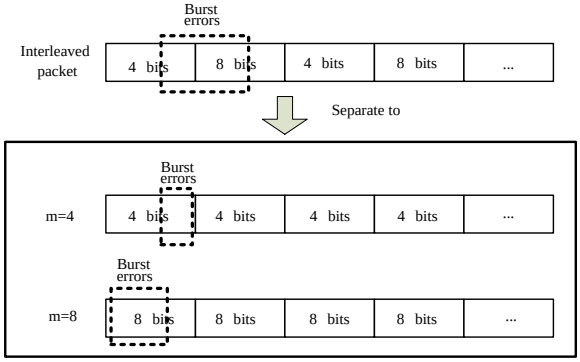


Figure 5. Burst errors dispersed on two symbols

III. ANALYTICAL MODEL

Two analytical models are built for comprehensive numerical simulations. The first one is referred to as uniform bit error model (UBEM), while the second one is referred to as on-off bit error model (OBEM). The first model assumes the errors occur evenly on the coded packets, while the second model assumes the errors may occur continuously in a burst length.

A. UBEM

Let P_{UB_be} and P_{UB_se} denote the probability of bit errors and the probability of symbol errors, respectively. Since any bit errors occur in a symbol may result in a symbol error and each symbol has m bits, we can derive P_{UB_se} directly from P_{UB_be}, as shown in Eq. (1).

$$P_{UB\_se} = 1-(1-P_{UB\_be})^{m} \quad (1)$$

Next, let us define two more parameters, N_S and N_C. The first parameter denotes the number of symbols in a codeword and the second parameter denotes the number of codeword in a packet. Hence, the uncorrectable probability of a codeword (P_{UB_cuc}), can be derived as shown in Eq. (2).

$$P_{UB\_cuc} = 1-\sum_{i=0}^{t}\left(\binom{N_s}{i}\times\left(P_{UB\_se}\right)^{i}\times\left(1-P_{UB\_se}\right)^{N_S-i}\right) \quad (2)$$

In Eq. (2), we know in a codeword if the number of symbol errors is smaller than t, then the codeword is correctable. Thus, the uncorrectable probability of a codeword can be summed up from $i=0$ to t, since there are $\binom{N_s}{i}$ different types of errors. Next, let us define P_{UB_puc} as the packet uncorrectable probability. Since there are N_C codeword in a packet, we can derive P_{UB_puc} as shown in Eq. (3).

$$P_{UB\_puc} = 1-\left(1-P_{UB\_cuc}\right)^{N_C} \quad (3)$$

B. OBEM

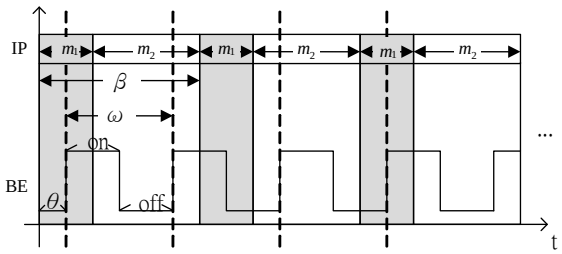


Figure 6. On-off bit error model

The on-off bit error model (OBEM) is illustrated in Figure 6. All the parameters used in the analysis are defined in Table I. A burst error (BE) period is defined as two consecutive bit error intervals where high bit errors appear first and then followed by low bit errors. Notice that θ_0 is defined as the length of right-shift position for an initial BE period; $\theta_0 = 0$ implies that no gap exists between the beginning of an interleaved packet and the beginning of the first BE period.

TABLE I. PARAMETERS USED IN OBEM

m_1	Number of bits in a symbol of HRR packet
m_2	Number of bits in a symbol of non-HRR packet
β	Packet interleaved length in bits ($m_1 + m_2$)
on	Length of high bit errors (in bits)
σ	Error probability of high bit errors
off	Length of low bit errors (in bits)
ρ	Error probability of low bit errors
ω	Burst length ($on + off$) (in bits)
θ_0	Length of initial right-shift position (in bits)

First, we define P_{OB_se} as the symbol-error probability in OBEM. An interleaved packet (IP) and a burst error (BE) may have different lengths; here we assume the former has a length of β bits ($\beta = m_1 + m_2$) and the later has a length of ω bits ($\omega = on + off$). Since every symbol in an IP may encounter different positions of bit errors, we have to analyze the bit error positions of a symbol before we can compute the symbol-error probability. To compute the error probability of the α^{th} symbol, we define (i) m_{1s} = the distance between the first bit of m_1 and the first bit of an IP, and (ii) m_{1e} = the distance between the last bit of m_1 and the first bit of an IP. Similarly, we define m_{2s} and m_{2e} for m_2. Thus, m_{1s}, m_{1e}, m_{2s}, and m_{2e} can be computed as shown in Eq. (4), (5), (6), and (7), respectively.

$$m_{1s} = \alpha \times \beta \quad (4)$$

$$m_{1e} = \alpha \times \beta + m_1 - 1 \quad (5)$$

$$m_{2s} = \alpha \times \beta + m_1 \quad (6)$$

$$m_{2e} = (\alpha + 1) \times \beta - 1 \quad (7)$$

For simplicity, the four parameters, m_{1s}, m_{1e}, m_{2s}, and m_{2e} are generalized to m_{ij}, where $i = 1, 2$ and

$j = s, e$. Let θ denote the length of right-shift position between an IP and a BE at the α^{th} symbol. Let $\Omega_{m_{ij}}$ denote the length of right-shift position for m_{1s}, m_{1e}, m_{2s}, and m_{2e} at the α^{th} symbol. Thus, we can compute θ and $\Omega_{m_{ij}}$ as shown in Eq. (8) and (9), respectively.

$$\theta = \left(\theta_0 + \left(\left\lceil \frac{header\ length}{\omega} \right\rceil \times \omega - header\ length\right)\right) mod\ \omega \quad (8)$$

$$\Omega_{m_{ij}} = \left(\left\lfloor \frac{m_{ij}+1}{\omega} \right\rfloor - 1\right) \times \omega + \theta \quad (9)$$

After we found the right-shift position between an IP and a BE, we can categorize the symbol errors into four cases. Case 1 shows whether or not a symbol may occupy one BE or two BE periods, their start bit and stop bit of a symbol all appear at the high-bit-error interval. Case 2 shows only the start bit of a symbol appears at the high-bit-error interval. Case 3 shows whether or not a symbol may occupy one BE or two BE periods, neither the start bit nor the stop bit appear at the high-bit-error interval. Finally, Case 4 shows only the stop bit of a symbol appears at the high-bit-error interval. Actually, the four different cases of symbol errors can be constrained by eight inequalities with four parameters, m_{ij}, $\Omega_{m_{ij}}$, on, and ω, as shown in Table II.

TABLE II. FOUR CASES OF SYMBOL ERRORS

Case	Conditions
1	$\Omega_{m_{is}} \le m_{is} < \Omega_{m_{is}} + on$ and $\Omega_{m_{ie}} \le m_{ie} < \Omega_{m_{ie}} + on$
2	$\Omega_{m_{is}} \le m_{is} < \Omega_{m_{is}} + on$ and $\Omega_{m_{ie}} + on \le m_{ie} < \Omega_{m_{ie}} + \omega$
3	$\Omega_{m_{is}} + on \le m_{is} < \Omega_{m_{is}} + \omega$ and $\Omega_{m_{ie}} + on \le m_{ie} < \Omega_{m_{ie}} + \omega$
4	$\Omega_{m_{is}} + on \le m_{is} < \Omega_{m_{is}} + \omega$ and $\Omega_{m_{ie}} \le m_{ie} < \Omega_{m_{ie}} + on$

Once we identify the four cases of symbol errors, we can compute the symbol-error probability. Since one codeword consists of N_s symbols, we define $P_{OB\_se}(\alpha), \alpha = 0,1,..., N_s - 1$, as the symbol-error probability of the α^{th} symbol. Let σ denote the error probability of high bit errors and ρ denote the error probability of low bit errors. Let N_σ represent the number of bits with errors in a symbol and N_ρ represent the number of bits without errors in a symbol. $P_{OB\_se}(\alpha)$ is computed as shown in Eq. (10).

$$P_{OB\_se}(\alpha) = 1 - (1-\sigma)^{N_\sigma} \times (1-\rho)^{N_\rho} \quad (10)$$

Now, we can compute N_σ and N_ρ for case 1 as shown in Eq. (11) and (12), case 2 as shown in Eq. (13) and (14), case 3 as shown in Eq. (15) and (16), and case 4 as shown in Eq. (17) and (18), respectively.

$$N_\rho = \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) - \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times Off$$
$$= \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor\right) \times Off \quad (11)$$

$$N_\sigma = m_i - N_\rho = m_i - \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor\right) \times Off \quad (12)$$

$$N_\sigma = \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) - \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times On + \Omega_{m_{is}} + On - m_{is}$$
$$= \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor\right) \times On + \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times \omega + \theta + On - m_{is}$$
$$= \left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor \times On + \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times Off + \theta - m_{is} \quad (13)$$

$$N_\rho = m_i - N_\sigma$$
$$= m_i - \left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor \times On - \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times Off - \theta + m_{is} \quad (14)$$

$$N_\sigma = \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) - \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times On$$
$$= \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor\right) \times On \quad (15)$$

$$N_\rho = m_i - N_\sigma = m_i - \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor\right) \times On \quad (16)$$

$$N_\rho = \left(\left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) - \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) - 1\right) \times Off + \Omega_{m_{is}} + \omega - m_{is}$$
$$= \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times Off + \left(\left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor - 1\right) \times \omega + \theta + \omega - m_{is}$$
$$= \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor \times On + \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) \times Off + \theta - m_{is} \quad (17)$$

$$N_\sigma = m_i - N_\rho = m_i - \left\lfloor \frac{m_{is}+1}{\omega} \right\rfloor \times On - \left(\left\lfloor \frac{m_{ie}+1}{\omega} \right\rfloor - 1\right) \times Off - \theta + m_{is} \quad (18)$$

By substituting N_σ and N_ρ back to Eq. (10), we can derive $P_{OB\_se}(\alpha)$ for the four different cases of symbol errors. After we compute the symbol-error probability for the four different cases, our next step is to derive the uncorrectable probability of a codeword and the uncorrectable probability of a packet. We know there are N_C codeword in a packet and the uncorrectable probabilities of the N_C codeword are all different. Let $P_{OB\_cuc}(l), l = 0,1,..., N_c - 1$, denote the uncorrectable probability of the l^{th} codeword and let P_{OB_puc}

denote the uncorrectable probability of a packet. By using the combination theory of probabilities, we can derive $P_{OB\_cuc}(l)$ and P_{OB_puc} as shown in Eq. (19) and (20).

$$P_{OB\_cuc}(l) = 1 - \sum_{i=0}^{t}\left(\sum_{j=0}^{\tau_i - 1}\left(\prod_{k=0}^{\lambda-1}\left(\binom{N_k}{r_k^{ij}} \times \left(P_{OB\_se}^{k}\right)^{r_k^{ij}} \times \left(1 - P_{OB\_se}^{k}\right)^{N_k - r_k^{ij}}\right)\right)\right) \quad (19)$$

$$P_{OB\_puc} = 1 - \prod_{l=0}^{N_C - 1}\left(1 - P_{OB\_cuc}(l)\right) \quad (20)$$

IV. NUMERICAL RESULTS

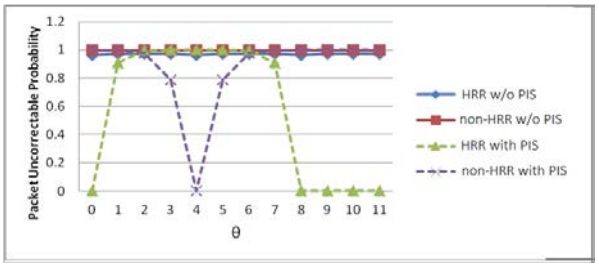
To study the influences of the four parameters, (i) the length of high bit errors (the on period), (ii) the length of low bit errors (the off period), (iii) the right-shift position (θ), and (iv) the number of correctable symbols (t), we perform numerical simulations. Table III shows the parameters and their setting used in the simulation.

TABLE III. PARAMETER SETTINGS

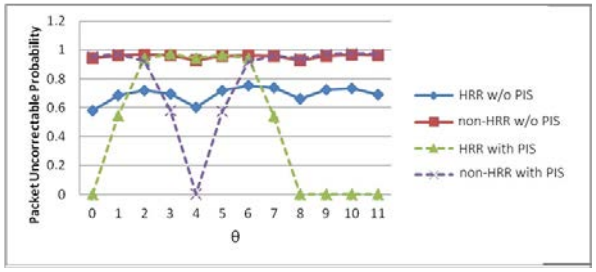
Parameter	Setting
m_1	4 bits
m_2	8 bits
on	4/6/8 bits
off	8/6/4 bits
σ	0.1
ρ	0.0001

A. Impact of Correctable Symbols

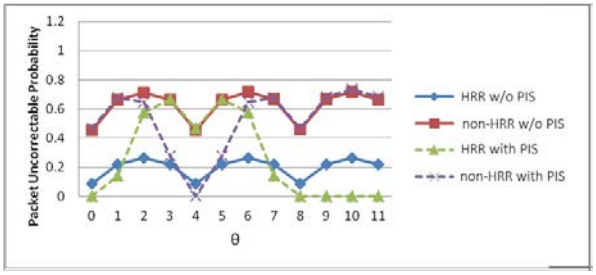
First, we are interested in studying the impact of increasing the number of correctable symbols when the length of high bit errors is shorter than the length of low bit errors; i.e., on = 4 bits and off = 8 bits. From Figure 7, we can observe that both the packet uncorrectable probabilities of HRR and non-HRR curves drop off very quickly, when the number of correctable symbols (t) is increased from 1 to 3. Additionally, we can observe that the curves of P_{OB_puc} with PIS (the two dashed lines) are much lower than the curves of P_{OB_puc} without using PIS (the two solid lines). The improvement of P_{OB_puc} by using PIS is more significant when t is small, which is quite beneficial for reducing packet overhead, since the number of redundancy bits in RS codes can be shorter. Another noticeable phenomenon is that although when θ is smaller than 7, HRR with PIS are completely inverted to non-HRR with PIS, the curves of HRR with PIS do drop to zero when θ is larger than 7.



(a) (on, off, t) = (4, 8, 1)



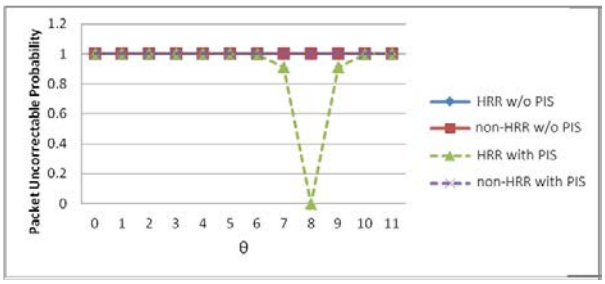
(b) (on, off, t) = (4, 8, 2)



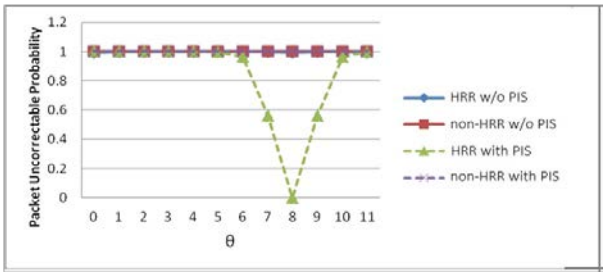
(c) (on, off, t) = (4, 8, 3)

Figure 7. Packet uncorrectable probability (on is smaller than off)

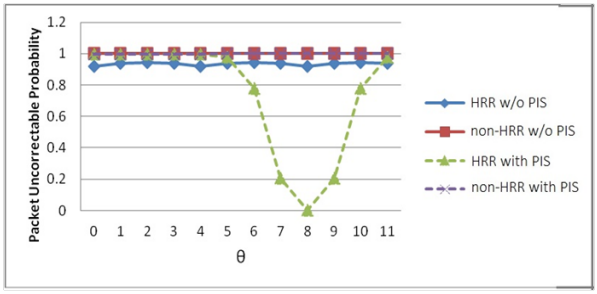
From Figures 8(a) to 8(c), we show the packet uncorrectable probabilities when the on period (8 bits) is longer than the off period (4 bits). It is interesting to notice that when t is larger than 3 and θ is smaller than 6, the packet uncorrectable probabilities of HRR with PIS are higher than those curves without PIS. Of course, when t is smaller than 3 and θ is larger than 6, the situations are completely inverted. Hence, from Figure 11, we have discovered that when the period of high bit errors exceeds the length of an HRR symbol (m = 4) and approaches to a non-HRR symbol (m = 8), it is better to encode packets with a small value of t; otherwise, there is no advantage achieved by using the proposed PIS.



(a) (on, off, t) = (8, 4, 1)



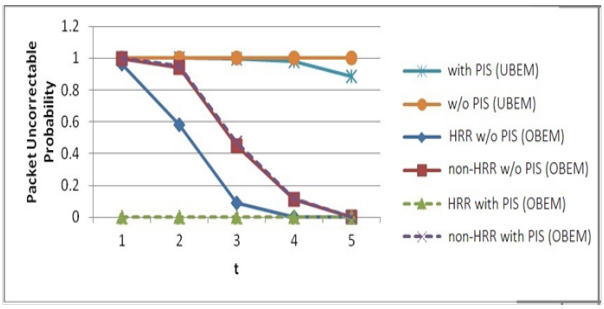
(b) (on, off, t) = (8, 4, 2)



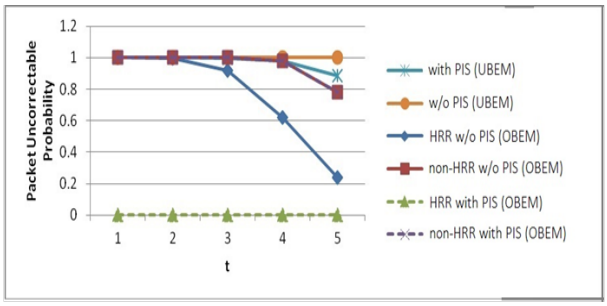
(c) (on, off, *t*) = (8, 4, 3)

Figure 8. Packet uncorrectable probability (on is larger than off)

Figures 9 shows the comparisons in packet uncorrectable probabilities between UBEM and OBEM. Notice that the curves of UBEM and the curves of OBEM vary along with the following two parameters: (i) when the number of correctable symbols increases from 1 to 5, the curves of UBEM dissever very quickly from those of OBEM; and (ii) when the right-shift position increases from zero to 8 bits, the gap between these two models becomes smaller. Since OBEM is more reactive to a real word than UBEM, it is rewarding to know that the proposed PIS can reduce packet uncorrectable probability in OBEM more significantly than that in UBEM. Another noticeable result is that no matter how we increase the period of high bit errors (the on period from 4 bits in 9(a) to 8 bits in 9(b)), HRR with PIS in OBEM always exhibits the lowest packet uncorrectable probability (near zero, in some cases). The relatively lower packet uncorrectable probability for HRR packets has demonstrated that the proposed PIS can successfully protect HRR packets from burst errors, while at the same time it does not sacrifice non-HRR packets from large uncorrectable bit errors.



(a) (on, off, θ) = (4, 8, 0)



(b) (on, off, θ) = (8, 4, 8)

Figure 9. Packet uncorrectable probability in UBEM vs in OBEM

V. CONCLUSIONS

In this paper, we have presented a packet interleaving scheme to reduce packet uncorrectable probability under burst errors in WSN. From the simulations, we have demonstrated that, no matter how we adjust the period of high bit errors, the proposed PIS behaves more resilient to burst errors in OBEM than in UBEM. Finally, by carefully adjusting the period of high bit errors and the right-shift positions, the PIS can reduce the uncorrectable probability of HRR packets to near zero.

REFERENCES

[1] F. Barac, K. Yu, M. Gidlund, J. Akerberg, and M. Bjrkman, "Towards Reliable and Lightweight Communication in Industrial Wireless Sensor Networks," INDIN2012: IEEE 10th International Conference on Industrial Informatics, Beijing, China, Jul. 25-27, 2012.

[2] S. Marinkovic and E. Popovici, "Network Coding for Efficient Error Recovery in Wireless Sensor Networks for Medical Applications," 2009 First Int'l Emerging Network Intelligence, Sliema, Malta, Oct. 11-16, 2009.

[3] B. Sklar, Digital Communications: Fundamentals and Applications, 2nd Ed. New Jersey, USA, 2001.

[4] H. Chebbo, S. Abedi, T. A. Lamahewa, D. B. Smith, D. Miniutti, and L. Hanlen, "Reliable Body Area Networks Using Relays: Restricted Tree Topology," Int'l Conference on Networking and Communications, Maui, Hawaiian Island, USA, Jan. 30 - Feb. 2, 2010.

[5] R. V. Sampangi, S. R. Urs, and S. Sampalli, "A Novel Reliability Scheme Employing Multiple Sink Nodes for Wireless Body Area Networks," 2011 IEEE Symposium on Wireless Technology and Applications (ISWTA), Langkawi, Malaysia, Sep. 25-28, 2011.

[6] S. Kim, R. Fonseca, and D. Culler, "Reliable Transfer on Wireless Sensor Network," First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, California, Oct. 4-7, 2004.

[7] E. Byrne, A. Manada, S. Marinkovic, and E. Popovici, "A Graph Theoretical Approach for Network Coding in Wireless Body Area Networks," 2011 IEEE International Symposium on Information Theory Proceedings (ISIT), Saint Petersburg, Russia, Jul. 31 - Aug. 5, 2011.

[8] Y. Hamada, K. Takizawa, and T. Ikegami, "Highly Reliable Wireless Body Area Network using Error Correcting Codes," 2012 IEEE Radio and Wireless Symposium, Santa Clara, CA, USA, Jan. 15-18, 2012.

[9] M. Luby, "LT Codes," The 43rd Annual IEEE Symposium on Foundations of Computer Science, Vancouver, British Columbia, Canada, Nov. 16-19 2002.

[10] K. Ishibashi, H. Ochiai, and R. Kohno, "Embedded Forward Error Control Technique for Low-Rate but Low Latency Communications," IEEE Trans. on Wireless Comm., vol. 7, no. 5, pp. 1456-1460, May 2008.

[11] M. Busse, T. Haenselmann, and W. Effelsberg, "Energy-Efficient Data Dissemination for Wireless Sensor Networks," Fifth IEEE Int'l Conference on Pervasive Computing and Communications Workshops, White Plains, New York, USA, Mar. 19-23, 2007.

[12] K. Yu, F. Barac, M. Gidlund, J. Akerberg, and M. Bjorkman, "A Flexible Error Correction Scheme for IEEE 802.15.4-based Industrial Wireless Sensor Networks," 2012 IEEE Int'l Symposium on Industrial Electronics (ISIE), Hangzhou, China, May 28-31, 2012.

[13] M. S. Srouji, Z. Wang, and J. Henkel, "RDTS: A Reliable Erasure-Coding Based Data Transfer Scheme for Wireless Sensor Networks," 2011 IEEE 17th International Conference on Parallel and Distributed Systems, Tainan, Taiwan, Dec. 7-9, 2011.

 ISBN: 978-1-61208-450-3

A Way of Eliminating Errors When Using Bloom Filters for Routing in Computer Networks

Gökçe Çaylak Kayaturan *, Alexei Vernitski †
Department of Mathematical Sciences,
University of Essex
Colchester, UK
Email: *gcayla@essex.ac.uk, †asvern@essex.ac.uk

***Abstract*—A Bloom filter is a data type for storing sets. It can be considered as a data compression technique, but its more important feature is an extremely fast access to stored data. This is why it can be useful when calculation needs to be performed very quickly, for example, in an application to routing messages in a computer network. A well-known shortcoming of a Bloom filter are errors in the stored data. We present a way of labelling links in a computer network which prevents errors in Bloom filters in some routing scenarios and, therefore, results in a more efficient use of network resources.**

***Keywords*—Bloom filter; computer network; routing.**

I. INTRODUCTION

A Bloom filter is a data type for storing sets. Its great advantages are space efficiency (that is, the space used is very small) and time efficiency (that is, accessing the stored data is extremely fast). This is why using Bloom filters has been suggested for a number of applications. A Bloom filter is probabilistic in the sense that it involves the use of pseudorandom hashes and, therefore, comes with a non-zero probability of errors called false positives [1]. This is why a general theme in the studies of Bloom filters is reducing the number of false positives.

Some applications of Bloom filters [2][3][4][5] to network problems, especially data mining, reducing data traffic and security of data transmission between parties, motivate us to introduce a new kind of encoding method for the items of the set represented by a Bloom filter. A wide range of research of network applications of Bloom filters was also presented in [6] and [7]. The study [8] proposes a Bloom filter protocol to minimize the memory for the storage of summary caches in order to reduce the web traffic. One variant of a Bloom filter called compressed Bloom filter [9] also focuses on decreasing the size of the network messages transmitted. For less computational requirements a path architecture has been established in [10] to increase the security of network traffic flow via Bloom filter. Secure transmission of messages is addressed in the studies of cryptographic Bloom filters [11] introducing a coding method for transmitting large data using Bloom filters. Sharing information between parties via grid models was proposed by [12] and [13].

The model we consider is a rectangular-grid computer network with messages forwarded between computers within it. Similar models were considered in [12][14]. We remove randomness from Bloom filters by constructing an explicit method of encoding links in the network. Our approach makes the use of Bloom filters more secure as well as optimizes the size of data to reduce the network traffic. The main advantage of our coding system is that it we encode sets using Bloom filters without introducing any errors at all.

This paper is organised with seven sections in total. In Section 2 the general definition of a standard Bloom filter is given, and optimal parameters are defined. In Section 3 we introduce a new encoding technique for edges in a rectangular grid network. In Section 4 we explain how our encoding method restricts and controls the number of 1s of Bloom filter. In Section 5 we prove that there are no false positives. In Section 6 we compare our encoding system with the performance of the standard Bloom filter. Section 7 is the conclusion.

II. STANDARD BLOOM FILTERS

A set S with n elements is represented by a binary array of m bits. In order to make this possible, each item which potentially can be an element of S is represented by an m bit array in which k bits are set to 1. The positions of these bits are chosen using pseudorandom hashes. All other bits are set to zero. We shall refer to these arrays as Bloom filters of the items. The Bloom filter of a set S is constructed by applying the bitwise OR operation to all the Bloom filters of the elements of S. Note that binary OR operation takes two bits as inputs and produces a bit 1 if at least one of the inputs is 1, or produces a bit 0 when all inputs are 0. This compression of the data of a set shows that the Bloom filter has space efficiency. After the Bloom filter of a set S has been constructed, one can query whether an item x belongs to the set S or not by comparing the Bloom filter of the set with the Bloom filter of x in all bit positions. If at one of these positions the Bloom filter of x is greater than the corresponding positions of the Bloom filter of S, then we conclude that the tested item is definitely not in the set S. However, when the tested item is less than or equal to the Bloom filter in all bit positions, then this item probably is in the set S. Some of these items, called 'false positives', are seemingly in the set S but are not really in the set. Namely, there is a probability that an item $x \notin S$ might be recognised by the Bloom filter to be in the set S, and then it is called 'false positive'. Even though some errors in the set occur, presence of any element in the set is checked very quickly. Hence comes the time efficiency of the Bloom filter.

 ISBN: 978-1-61208-450-3

An approximate value of the probability of false positives in a Bloom filter is usually expressed is a function which depends on three parameters k, m, n that are the number of 1s of items of the set S, length of these items and number of items of the set, respectively. The study of Bloom [1] shows that the probability of an item being a false positive is as given in the approximate formula (1), which is obtained by a simple probability argument.

$$\left(1 - (1 - \frac{1}{m})^{kn}\right)^k \approx \left(1 - e^{\frac{-kn}{m}}\right)^k \quad (1)$$

The probability of being a false positive was expressed in other ways by [6][8][9][15]. The smallest probability of a false positive is reached when $k = \ln 2 \times \frac{m}{n}$, as can be seen by taking the derivative of (1). In theory, the smallest probability may be obtained when k is exactly $\ln 2 \times \frac{m}{n}$, but in practice k must be an integer.

III. Encoding paths in a rectangular grid

The model we are considering is a computer network having a shape of a rectangular grid of size $M \times N$, that is, M links horizontally in each row and N links vertically in each column. We assume that there is a computer at each node of the grid. We assume that from time to time a computer in the network may need to send a message to another computer in the network; then the sender computer encloses a header with the message, which describes exactly what path the message must follow. For this purpose, each edge is allocated its own Bloom filter in advance, and the path is represented as the set of its edges, that is, the Bloom filter of the path is the bitwise OR of the Bloom filters of the edges constitution the path. We assume that only shortest paths in the grid are used to deliver messages between the computers.

Lemma 1. A directed shorthest path between two distinct vertices in a rectangular grid consists of directed edges with at most two different directions.

Proof: Consider an imaginary straight line between two distinct vertices in a rectangular grid. According to Euclid geometry this straight line is the shortest distance between two given vertices. But the path consists of the edges between the vertices. This imaginary line can be best approximated by directed edges pointing in only two different directions, and hence the shortest path includes only such edges. ■

We introduce two systems of coordinates for the grid: the one starting from the bottom-left corner and the second starting from the top-left corner of the grid. Accordingly, we introduce two notations for each vertex, with the letter u in the former system of coordinates and with the letter v in the latter.

Each edge is represented by a Bloom filter with the length $m = 4 \times (M + N)$ bits. The Bloom filter consists of two equal length parts, which correspond to the two systems of coordinates. The vertex at the bottom left corner of the grid is denoted by $u_{(0,0)}$ and is the origin of the x/y coordinate system, with the coordinates of the vertices increasing in the direction of north-east. The point $v_{(0,0)}$, the origin of the other x/y coordinate system, is the vertex at the top left corner of the grid, and the coordinates of vertices increase in the direction of south-east.

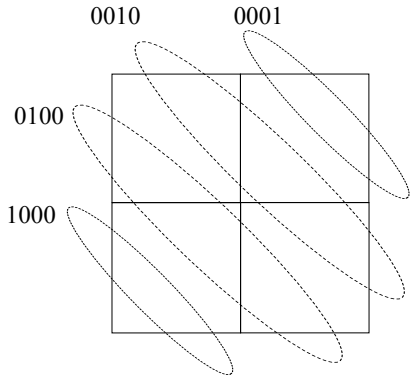


Figure 1. The allocation of the first halves of the Bloom filters of the edges in each cell.

Encoding an edge is based on the coordinates of the edges incidental with it and on the horizontal or vertical orientation of the edge in the grid. That is, when an edge is horizontal, the first and the second halves of the Bloom filter of the edge will be based on the vertex $u_{(i,j)}$, otherwise known as $v_{(i,N-j)}$, that is the left endpoint of the encoded edge.

However, if an edge is vertical, the vertices to encode it will be $u_{(i,j)}$, which is on bottom endpoint of the edge, and $v_{(i,N-j-1)}$, which is top endpoint of the same edge. Predictably, the first half of the Bloom filter of a vertical edge will be specified with the vertex $u_{(i,j)}$ and the second half will be constructed with $v_{(i,N-j-1)}$.

It is useful to imagine each half of the Bloom filter encoding an edge as a sequence of two-bit long blocks. There are $2(M+N)$ two-bit blocks in total in the Bloom filter, since each half of the encoded edges has $(M+N)$ blocks. Exactly one block of each half contains 1, and it is either 01 or 10 (as described further). All other blocks are 00. To encode a vertical edge, the bit 1 will be placed in the first positions of both $(i+j+1)$th and $(M+N+i+N-j-1+1)$th blocks in the first and the second half of the Bloom filter of the edge, respectively. To encode a horizontal edge, the bit 1 is placed in the second position in $(i+j+1)$th and $(M+N+i+N-j+1)$th blocks in the first and the second halves of Bloom filter of edge, respectively.

Let us look at an example. The first halves of the Bloom filters of the edges of a 2×2 grid are presented in Figure 1, and the second halves of their Bloom filters are presented in Figure 2. Consider every cell of both Figure 1 and Figure 2 as a rectangular grid with size 1×1. A horizontal edge of a cell is represented with the points $u_{(0,0)}$ and $v_{(0,1)}$, which are the left endpoint of the edge, in a rectangular grid with size 1×1. Then the block 01 will be placed in the $0+0+1 = 1$st and $1+1+0+1+1 = 4$th block positions among $2(1+1) = 4$ blocks. Note that the length of the Bloom filters of edges is $4(1+1) = 8$ bits. So the Bloom filter of this horizontal edge will look like 01000001. Besides, a vertical edge is represented with the points $u_{(0,0)}$ and $v_{(0,0)}$, which are the bottom and top endpoints of the vertical edge, respectively, in the grid with size 1×1. The block 10 will be situated in the $0+0+1 = 1$st and $1+1+0+0+1 = 3$th block positions. The Bloom filter

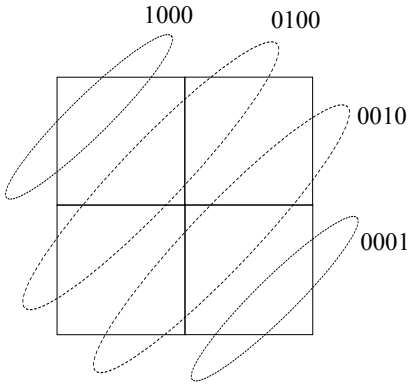


Figure 2. The allocation of the second halves of the Bloom filters of the edges in each cell.

of the this vertical edge will be as 10001000.

IV. An analysis of Bloom filters of paths

A. Positions of 1s in the Bloom filters

All possible shortest paths consist of $n \geq 1$ edges lying between two distinct vertices continuously. All Bloom filters of edges include two 1s and the Bloom filter of a path is obtained by applying OR operation to the encoded adjacent edges lying on the path together.

Lemma 2. Consider an undirected path whose direction it towards north-east (or, equivalently, south-west). Then the first half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form 01 and 10.

Likewise, if we consider an undirected path whose direction it towards south-east (or, equivalently, north-west), then the second half of the Bloom filter of the path contains a consecutive subsequence of blocks having the form 01 and 10.

Proof: This is because the coordinates of vertices u of the path edges are increasing towards north-east and the blocks in the first halves of the Bloom filters of the edges are specified by the points of u. Likewise, if we consider the second halves of Bloom filters of edges of a path directed towards south-east, the coordinates of vertices v increase in the same way. ■

One can observe that some blocks in the second half of the Bloom filter of a path going north-east contain two bits 1 at the same time. Similarly, the first half of the Bloom filter of a paths going south-east might include some blocks 11. If all the edges of a specific path are turned in one direction (north or south or west or east), then all blocks which include one bit 1 in either half of the Bloom filter of the path come after each other consecutively without an interruption of a block such as 00 or 11.

B. The number of 1s in the Bloom filters

In our encoding the number of 1s in the Bloom filter of each encoded edge is 2 and they are not placed randomly but depend on the position of the edge. The Bloom filters of shortest paths in a rectangular grid are obtained without randomness being involved, hence, the number of 1s in these Bloom filters is restricted.

Lemma 3. The number of 1s of the Bloom filter of the paths in rectangular grid is not greater than $2n$ where the number of edges of the path is n.

Proof: By the lemma 2, the blocks 10 or 01 constitute a continuous sequence of blocks within the first or the second half of the Bloom filter of a path. So, these blocks 10 or 01 represents the number of edges in the path and there are n edges in total in a path. Hence, when the number of the bits 1 in one half of the Bloom filter of the path is n, the number of bits 1 will be $\leq n$ in the other half of the Bloom filter. Any half of Bloom filter of path does not necessarily have n 1s, since the 1s in one half of the Bloom filter of a path might be produced in the same positions by more than one edge. More precisely, the path might contain the edges that are encoded on the same block position with the same pair bits including 1. Therefore, the bits 1s of the Bloom filter of these paths are $\leq 2n$.

We may note that if the edges of a path are directed in only one way, then both halves of the Bloom filter of the path will contain n 1s. ■

V. Avoiding false positives

Theorem 4. All Bloom filters of edges are unique in a rectangular grid.

Proof: All edges of a grid of size $M \times N$ are encoded by $2(M+N)$ blocks and each block consists of two bits. Suppose two edges e and f are distinct and both are horizontal. So, the 1s of these edges will take place in second bit positions in corresponding blocks.

The vertices of the horizontal edge e will be $u_{(i,j)}$ and $v_{(i,N-j)}$ and the vertices of the horizontal edge f will be given with $u_{(k,l)}$ and $v_{(k,N-l)}$, when we accept the left top and bottom corners of the grid as the centres of two different x/y coordinate systems. Both halves of horizontal edges are encoded with the left endpoints of the edges. The blocks containing 1s will be placed in $(i+j+1)$th and $(M+N+i+N-j+1)$th block positions of the Bloom filter of edge e. Similarly, $(k+l+1)$th and $(M+N+k+N-l+1)$th blocks of the Bloom filter of edge f contain 1s.

Assume the edges f and e are represented by the same Bloom filter. That means both $i+j+1 = k+l+1$ and $M+N+i+N-j+1 = M+N+k+N-l+1$ occur at the same time.

$$i + j = k + l \tag{2}$$

$$i - j = k - l \tag{3}$$

Then, the equations $i = k$ and $j = l$ are obtained. This contradicts with the assumption that the edges e and f are distinct.

If we suppose that both edges e and f are vertical, then both edges will contain 1s on the first positions of the corresponding blocks. Assume these two edges are distinct and encoded by the same Bloom filter. Then the same equations concerning the coordinates of the vertices $u_{(i,j)}$, $v_{(i,N-j-1)}$ and $u_{(k,l)}$, $v_{(k,N-l-1)}$ as above can be constructed and obviously the

 ISBN: 978-1-61208-450-3

same contradiction which is found for the edges lied horizontally will be obtained.

Now we suppose that one of the distinct edges is horizontal and the other is vertical. Namely, when e is vertical, then the bit 1 of the Bloom filter of e will only appear in the first positions of some blocks. Yet the bit 1 of the Bloom filter of the edge f will appear in the second position of some blocks, since f is horizontal. Hence even if the blocks including 1s of these two distinct edges are in the same block positions, these 1s are not placed on the same positions.

As a result of considering these three cases, we conclude that all edges of a grid are encoded uniquely in a given size grid. ■

Theorem 5. The Bloom filter of a path consisting of $n \geq 1$ edges in a rectangular grid model does not yield any false positives when links adjacent to the path are queried.

Proof: In the proof we shall concentrate on one fixed node of a path and demonstrate that no more than one link will be recognised by its Bloom filter as the next link of the path.

Consider the Bloom filter of a shortest path in a rectangular $M \times N$ grid. Suppose a message travelling along this path has arrived to some node via an edge e. It can continue travelling via any of the next three edges adjacent to e at one end, and the computer at the node needs to decide which one of them to choose. We shall see that the Bloom filter of the path $\beta(P)$ including edge e does not yield any false positives, that is, only one of these three edges is recognised.

An edge e encoded with $4(M+N)$ binary bits is represented by $\beta(e)$ that is divided into two bits length blocks called $\beta_1(e), \beta_2(e), \ldots, \beta_{2(M+N)}(e)$. The two individual bits constituting a block $\beta_p(e)$ will be denoted by $\beta_p^1(e)$ and $\beta_p^2(e)$. Note that the Bloom filters of the edges are divided into two equal parts and the positions of the blocks containing 1s of all Bloom filters of edges are specified with the coordinates u for the first half and coordinates v for the second half. When an edge is horizontal, the bit 1 is in the second positions of certain two blocks in the first and the second half of the Bloom filter, and the remaining blocks are all 00. The positions of 1s are determined by the coordinates $u_{(i,j)}$ and $v_{(i,N-j)}$ and are at the positions $\beta^2_{(i+j+1)}(e)$ and $\beta^2_{(M+N+i+N-j+1)}(e)$, where $(i+j+1)$ and $(M+N+i+N-j+1)$ represent the positions of the blocks. Note that we will use k instead of $(i+j+1)$ and l instead of $(M+N+i+N-j+1)$ to make the indices simpler.

If the edge e is vertical, then 1s of the blocks will be in the first positions of the corresponding blocks. These bits will be at positions $\beta^1_k(e)$ and $\beta^1_{l-1}(e)$, since the endpoint vertices of a vertical edge e are $v_{(i,j)}$ and $u_{(i,N-j-1)}$.

Let the three edges adjacent to the edge e at one end be f, g and h (as shown on Figure 3). We already know that $\beta(e) \neq \beta(f) \neq \beta(g) \neq \beta(h)$. Now, we will look for the false positives of the Bloom filters of the paths for these four possible directions.

Firstly, we will suppose that the edge e is on the path where the message moves east and the represented vertices of e will be $u_{(i,j)}$ and $v_{(i,N-j)}$ (see Figure 3).

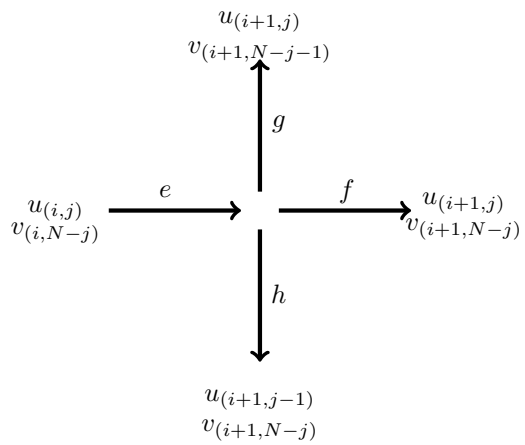


Figure 3. The edge e is supposed to lie on a path which is on the way of east and the next three adjacent edges of e.

The next edge the the message can follow after the edge e can be the one to the east, north or south, that is, the edges f, g, h, which are represented by the vertices $u_{(i+1,j)}$ and $v_{(i+1,N-j)}$, $u_{(i+1,j)}$ and $v_{(i+1,N-j-1)}$, $u_{(i+1,j-1)}$ and $v_{(i+1,N-j)}$, respectively. By the encoding method, we add the values of the points of represented vertices and 1 together to find the block positions of the pairs including 1s of the edges. So, the bits $\beta^2_k(e)$ and $\beta^2_l(e)$ will be 1. As a result of the encoding method, the bits $\beta^2_{(k+1)}(f)$ and $\beta^2_{(l+1)}(f)$, $\beta^1_{(k+1)}(g)$ and $\beta^1_{(l)}(g)$, $\beta^1_{(k)}(h)$ and $\beta^1_{(l+1)}(h)$ will be 1.

Note that since the edge e lies on the path, the bits of corresponding blocks of the Bloom filter of the path are $\beta^2_{(k)}(P) = 1$ and $\beta^2_l(P) = 1$.

The proof proceeds by considering several cases: that f is on the path, that g is on the path and that h is on the path.

If both $\beta^2_{(k+1)}(f) \leq \beta^2_{(k+1)}(P)$ and $\beta^2_{(l+1)}(f) \leq \beta^2_{(l+1)}(P)$ occur at the same time, then we say whether the path includes the edge f or f is a false positive of the path. But the bit of the path $\beta^1_{(k+1)}(P)$ is 0. Since if the path goes north-east, then the first half of the Bloom filter of path will contain the blocks which consist of one 0 and one 1 consecutively (see lemma 2). That means $\beta^1_{(k+1)}(g) = 1 > \beta^1_{(k+1)}(P) = 0$. Namely, the Bloom filter of the path is not greater than or equal to the Bloom filter of the edge g in all bit positions. As a result, when the Bloom filter of the edge f is smaller than or equal to the Bloom filter of the path P in all bits positions, more precisely $\beta(f) \leq \beta(P)$, then the edge g is definitely not on the path.

When both Bloom filters of e and f are smaller than or equal to the Bloom filter of path in all bits positions at the same time, then the two consecutive blocks $\beta_k(P)\beta_{(k+1)}(P)$ of the Bloom filter of the path will look like 0101. This is because the coordinates of the vertices used to produce the blocks including the bit 1 in the first half of the Bloom filter are increasing towards north-east, and the edges e and f are followed along the path towards the east and north. Under these circumstances, when the edge e lies on the path and $\beta(f) \leq \beta(P)$, then $\beta^1_{(k)}(P) = 0$. But $\beta^1_{(k)}(h) = 1 > \beta^1_{(k)}(P) = 0$, then we conclude that the edge h is not on path. The Bloom filter of

edge h is not smaller than or equal to the Bloom filter of the path in all bits positions.

Now, let us consider another case, if $\beta(g) \leq \beta(P)$ in all bits positions, then we say the edge g lies on the path after the edge e or a false positive of the Bloom filter of the path. Again, when both Bloom filters of e and g are smaller than or equal to the Bloom filter of the path in all bits positions and assume the direction of travel towards the east and north, which are the ways that make the blocks including one 1 and one 0 in the first part of the Bloom filter of the path to lie consecutively. So $\beta^2_{(k+1)}(P) = 0$, but we obtain that $\beta^2_{(k+1)}(f) > \beta^2_{(k+1)}(P)$, and hence the edge f is not on the path. Subsequently, by comparing the first bit of the (k)th block in the Bloom filter of the path and the Bloom filter of the edge h, we will conclude that h is definitely not an edge of the path.

Hence, both $\beta(e) \leq \beta(P)$ and $\beta(g) \leq \beta(P)$ occur at the same time, then both $\beta(f)$ and $\beta(h)$ are not smaller than or equal to the Bloom filter in all bits positions.

Finally, if both $\beta(e) \leq \beta(P)$ and $\beta(h) \leq \beta(P)$ in all bits positions, then the second part of the Bloom filter of the path will contain the blocks consisting of one 0 and one 1, consecutively. The coordinates of vertices v increase towards south-east and e and h can be travelled towards east and south, respectively. But $\beta^2_{(l+1)}(f) > \beta^2_{(l+1)}(P)$ and $\beta^1_{(l)}(g) > \beta^1_{(l)}(P)$. Hence, when $\beta(h) \leq \beta(P)$, the edges f and g are definitely not on the path.

As a result, the edge e is followed by exactly one of the adjacent edges f or g or h. As shown above, when e lies on the path and any adjacent edges of e is smaller than or equal to the Bloom filter in all bits positions at the same time, then the other adjacent edges are definitely not on the path.

We considered the situation when the edge e has been travelled by the message eastwards. When the edge e is oriented some other way, so that message travels along it north or west or south, the same argument as above can be applied. ■

VI. The advantages of our approach

The standard Bloom filter is defined as a randomized data structure. If our model included randomness, we would certainly get some false positives. However, we use the standard Bloom filter approach (indeed, we use disjunction and comparison of binary arrays precisely in the way as it is done, when the Bloom filters are used) without randomness. The possible false positives of our use scenario are the edges adjacent to a fixed path which the messages must follow. If false positives existed, the messages would be sent along these adjacent edges instead of following only the specified path.

If one position per edge was used in the Bloom filter (in a naive attempt to avoid false positives), the length of the Bloom filter would be $2MN + N + M$, which is the total number of the edges of $M \times N$ sized grid. Yet the length of bloom filter in our model is $4(M + N)$, which is better, since $4(M + N) < 2MN + N + M$ for the big values of M and N.

In our model, the number of edges lying on a path takes its maximum value with $(M+N)$, when the two distinct vertices are the two opposite corners of the rectangular grid. Therefore, the number of edges on a path is $\leq (M+N)$. The length of all encoded edges and the Bloom filter of any path is a constant $4(M + N)$.

Let us see how many false positives the standard Bloom filter would produce if we used it to represent sets of this size. The formula for the optimal value of the number of the bits 1 of an item is obtained as $k = \ln 2 \times \frac{m}{n}$, when the probability of the false positives is minimized, [9]. The ratio of the length of the Bloom filter and the number of edges of the path of our model is $\frac{m}{n} = \frac{4(M+N)}{(M+N)} = 4$, where n takes its maximum value. Hence, the number of 1 of an edge is $k = \ln 2 \times 4 \approx 2,772$. Hence, in order to obtain the minimal probability of the false positive of the Bloom filter of the path, the number of 1 of the all Bloom filters of the edges can be chosen to be $k = 2$, like in our model.

When the number of edges on the path is maximal, then the probability of false positives of the Bloom filter of the path would have been obtained as (4).

$$\left(1 - e^{\frac{-2(M+N)}{4(M+N)}}\right)^2 \approx 0,1548 \tag{4}$$

Of course, for some paths the number of edges on the path is less than $M+N$. For such shorter paths, the ratio $\frac{n}{m}$ decreases. Hence, on average the probability of false positives would be less than the equation (4). Hence, if the bits 1 would have taken place in the Bloom filter of the edges randomly, the probability of having no false positives in a path would be approximately $0.85^{2(M+N)}$, where $2(M + N)$ is the number of adjacent edges of the shortest path with maximum number of edges in the grid.

VII. Conclusion

This model uses active knowledge about the particular application of Bloom filters and combines space efficiency and time efficiency with one hundred percent accuracy by intelligently allocating Bloom filters to individual items. We shall continue this research by considering other network topologies.

References

[1] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," Communications of the ACM, vol. 13, no. 7, 1970, pp. 422–426.

[2] Y. Lu, B. Prabhakar, and F. Bonomi, "Perfect hashing for network applications," in Information Theory, 2006 IEEE International Symposium on. IEEE, 2006, pp. 2774–2778.

[3] C. E. Rothenberg, C. A. B. Macapuna, M. F. Magalhães, F. L. Verdi, and A. Wiesmaier, "In-packet bloom filters: Design and networking applications," Computer Networks, vol. 55, no. 6, 2011, pp. 1364–1378.

[4] L. Carrea, A. Vernitski, and M. Reed, "Yes-no bloom filter: A way of representing sets with fewer false positives for in-packet path encoding," 2014, submitted for publication.

[5] X. Yang, A. Vernitski, and L. Carrea, "An approximate dynamic programming approach for improving accuracy of lossy data compression by bloom filters," accepted, European Journal of Operational Research.

[6] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet mathematics, vol. 1, no. 4, 2004, pp. 485–509.

[7] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of bloom filters for distributed systems," Communications Surveys & Tutorials, IEEE, vol. 14, no. 1, 2012, pp. 131–155.

[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," IEEE/ACM Transactions on Networking (TON), vol. 8, no. 3, 2000, pp. 281–293.

 ISBN: 978-1-61208-450-3

[9] M. Mitzenmacher, "Compressed bloom filters," IEEE/ACM Transactions on Networking (TON), vol. 10, no. 5, 2002, pp. 604–612.
[10] T. Wolf, "A credential-based data path architecture for assurable global networking," in Military Communications Conference, 2007. MILCOM 2007. IEEE. IEEE, 2007, pp. 1–7.
[11] M. Mitzenmacher and G. Varghese, "Biff (bloom filter) codes: Fast error correction for large data sets," in Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on. IEEE, 2012, pp. 483–487.
[12] X. Li, L. Peng, and C. Zhang, "Application of bloom filter in grid information service," in Multimedia Information Networking and Security (MINES), 2010 International Conference on. IEEE, 2010, pp. 866–870.
[13] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on. IEEE, 2001, pp. 181–194.
[14] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," ACM SIGMOBILE mobile computing and communications review, vol. 7, no. 3, 2003, pp. 19–20.
[15] J. K. Mullin, "A second look at bloom filters," Communications of the ACM, vol. 26, no. 8, 1983, pp. 570–571.
[16] L. Carrea, A. Vernitski, and M. Reed, "Optimized hash for network path encoding with minimized false positives," Computer networks, vol. 58, 2014, pp. 180–191.

 ISBN: 978-1-61208-450-3

openwincon: Open Source Wireless-Wired Network Controller

Software Defined Infrastructure (SDI) approach for Fixed-Mobile-Converged Enterprise Networks

Gijeong Kim
Department of Computer Engineering
Kyung Hee University
South Korea
Email: kimgijeong@khu.ac.kr

Sungwon Lee
Department of Computer Engineering
Kyung Hee University
South Korea
Email: drsungwon@khu.ac.kr

***Abstract*— Software Defined Networking/Network Function Virtualization (SDN/NFV) is considered as a key technology for future mobile networks, such as 5G mobile communications systems. By 2020, the market size for Centralized-Radio Access Networks (C-RAN) will reach $21B. At the same time, enterprise Software Defined Data Center (SDDC) and the mobility market will increase to $77B and $360B, respectively. However, research on mobile enterprise networking is not widely considered. Also, open source software and hardware for an enterprise network is not well developed. Thus, in this paper, we introduce the *openwincon* project. *openwincon* is an open source wireless-wired network controller, and it is based on Software Defined Infrastructure (SDI) technology. *openwincon* is sponsored by the Korean government and started in 2015. The source code will be available in early 2016.**

Keywords-component; SDI; SDN; NFV; Enterprise Network.

I. INTRODUCTION

Software Defined Networking/Network Function Virtualization (SDN/NFV) is considered as a key technology for future mobile networks, such as 5G. Its world market size is estimated at $19B. By the year 2018, telecom SDN/NFV will be $11B and enterprise SDN/NFV will be $8B. This number is larger than C-RAN's estimated $10B. By 2020, when the 5G mobile service will start, C-RAN's market size will reach $21B, but enterprise SDDC and mobility market will increase to an estimated $77B and $360B, respectively [1].

According to SDN/NFV for public networks, many open source efforts are under way, such as Open Daylight (ODL) [2], Open Networking Operating System (ONOS) [3], and Open Platform for Network Function Virtualization (OPNFV) [4]. Conventional researches are focused on the evolution of these platforms. However, research on mobile enterprise networks is just at the beginning [5] and it is not widely considered. Also, open source software and hardware for an enterprise network is not well developed in general.

Thus, in this paper, we introduce the *openwincon* project. *openwincon* is an open source wireless-wired network controller, and it is based on Software Defined Infrastructure (SDI) technology. In this project, we propose a new networking architecture that converges wireless and wired networks through SDN/NFV. It provides a control-bearer separated enterprise network through a single centralized controller. *openwincon* is sponsored by the Korean government and started in 2015 and will last until 2020. The open source software code opens in early 2016.

II. PREVIOUS WORKS

The conventional heterogeneous network architecture is depicted in Figure 1. In the figure, a wired network (Giga Internet) and four different wireless networks are considered. Among the wireless networks, WiFi, 2G, 3G and 4G are considered. Each network has its wireless links, access networks, and core networks.

For this reason, centralized and integrated control is not possible. In addition, integrated QoS/QoE (Quality of Service/Quality of Experience) for high-level services are practically impossible. Moreover, the network complexity is increased in terms of network manageability and controllability.

However, SDN/NFV provides a solution for this type of environment. For example, the bearer planes for most networks are similar, and can be integrated into a single device. Also, a difference between networks can be implemented as a single physical device with virtualized applications.

III. PROPOSED *OPENWINCON* ARCHITECTURE

The mobile network architecture based on *openwincon* is depicted in Figure 2. We support conventional wired and wireless networks and future networks such as 5G. For this, not only Single Radio Cells (SRCs) but also Multiple Radio Cells (MRCs) are considered. MRC supports multiple

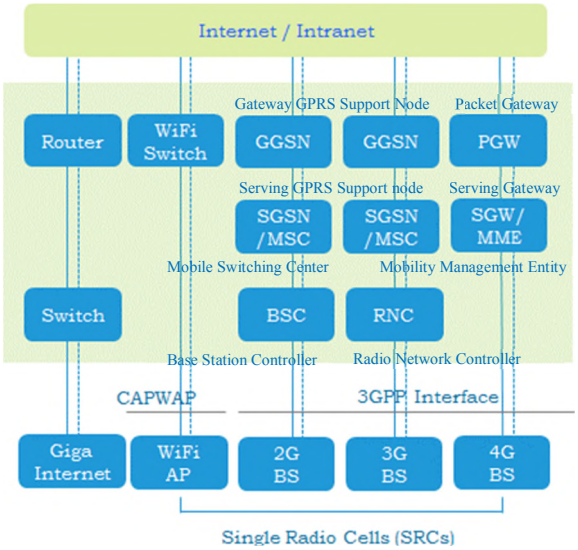


Figure 1. Conventional Heterogeneous Network Architecture

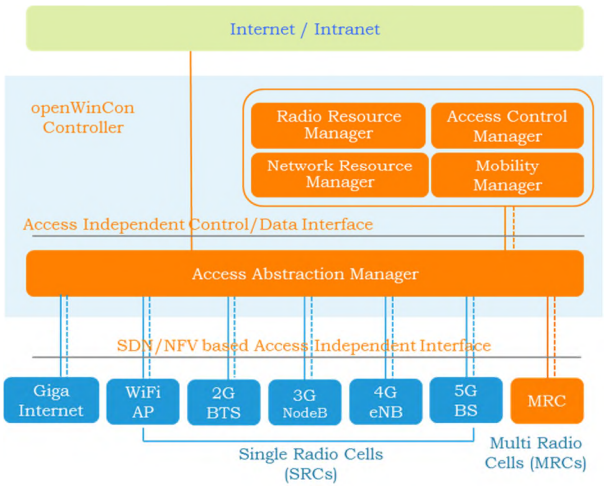


Figure 2. Proposed openwincon based Network Architecture

generation air interfaces simultaneously operating on the same physical device.

Five major features are considered. First, the Access Abstraction Manager (AAM) terminates air link dependent functionalities to abstract the multiple networks. This manager terminates protocols such as WiFi Control and Provisioning of Wireless Access Points (CAPWAP), and LTE S1. Second, the Radio Resource Manager (RRM) manages wireless resources such as frequency, bandwidth, and interference. Also, wireless scheduling is enabled to guarantee the QoS/QoE of services. Third, the Network Resource Manager (NRM) manages wireless resources. NRM includes OpenFlow-based network control and traditional traffic engineering. Fourth, the Access Control Manager (ACM) provides managements for subscribers, devices, and services. ACM includes control and management of authentication, authorization, accounting, security, and service policy. Fifth, the Mobility Manager (MM) provides mobility control for subscribers and devices. MM provides seamless and consistent handover scheme over heterogeneous wireless network environments.

We support carrier-grade reliability initially. As depicted in Figure 3, each controller supports reliability via a distributed multi-core architecture. Each manager operates over multiple distributed cores. It avoids service destruction when a single core fails. Also, inter-controller reliability is guaranteed through multi-site redundancy. Thus, a controller level failure is recovered via a peering controller.

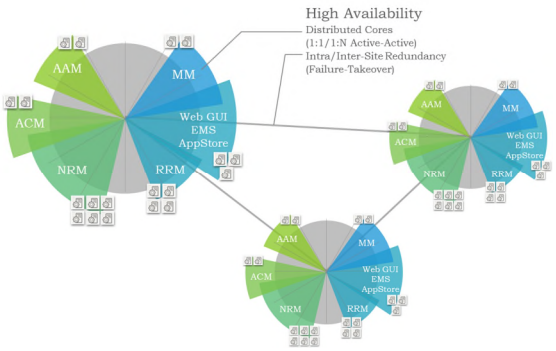


Figure 3. Redundancy and reliability support of openwincon

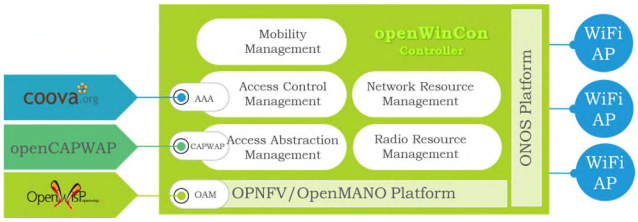


Figure 4. Proposed openwincon Internal Architecture for First Release

IV. CONCLUSIONS

Our project is sponsored by the Korean government for five years. Four universities participate: Kyung Hee University (KHU), Seoul National University (SNU), Pohang University of Science and Technology (POSTECH), and Sung Kyun Kwan University (SKKU). Also, SDN/NFV related companies and government offices participate in the project.

According to the development platform, we use ONOS SDN controller as a control plane platform and OPNFV as a bearer plane platform, as depicted in Figure 4. Figure 4 describes the goal of openwincon in the first year.

According to heterogeneous wireless technologies, we are going to re-use conventional open source solutions such as coova.org, openCapwap, OpenWISP, OpenWRT [6] for WiFi AP (Access Points), OpenBTS [7] for 2G/3G base stations, and Open-Air interface [8] for 4G.

The software source code will be released in early 2016. Apache 2.0 license will be applied. Also, the performance matrix will be announced with the source code.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-15-1366, Development of Core Technology for Autonomous Network Control and Management), (B0190-15-2013, Development of Access Technology Agnostic Next- Generation Networking Technology for Wired-Wireless Converged Networks), (B0101-15-0033, Research and Development of 5G Mobile Communications Technologies using CCN-based Multi-dimensional Scalability).

REFERENCES

[1] Mind Commerce, "Software Defined Networks (SDN) and Network Function Virtualization (NFV) Market, Forecasts, and Impact on Network Operators 2015 - 2020", March 2015.

[2] Open Daylight (ODL) official site, https://www.opendaylight.org, [Retrived: Feb., 2016].

[3] Open Network Operating System (ONOS) official site, http://onosproject.org, [Retrived: Feb., 2016].

[4] Open Platform for Network Function Virtualization (OPNFV) official site, https://www.opnfv.org, [Retrived: Feb., 2016].

[5] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao, "Towards Programmable Enterprise WLANs with Odin", HotSDN'12, August 13, 2012, pp.115-120.

[6] OpenWrt official site, https://openwrt.org, [Retrived: Feb., 2016].

[7] OpenBTS official site, http://openbts.org, [Retrived: Feb., 2016].

[8] Open Air Interface official site, http://openairinterface.org, [Retrived: Feb., 2016]

 ISBN: 978-1-61208-450-3

Function Oriented Network Architecture for Realization of Autonomic Networks

Gijeong Kim, Sungwon Lee*
Department of Computer Engineering
Kyung Hee University
Youngin-si, Korea
{kimgijeong, drsungwon}@khu.ac.kr

***Abstract*—Existing networks have focused on high data transfer rate and reliable data delivery through lower header processing and effective signaling. Recently, reducing CAPEX/OPEX (Capital Expenditures/Operating Expenditure), deploying new network services, and orchestration and management are issues of growing importance in the network. To solve these issues, we propose a Function Oriented Network (FON) that can facilitate the rapid deployment of new network functions and protocols to satisfy the requirements that are generated continuously over time. In this work, we introduce the FON architecture and describe the implementation of the FON prototype. Through this implementation, we demonstrate flexible network programmability and support core architecture to realize autonomic networking.**

Keywords-Autonomic Networks; Bio-inspired Networking; Network Programmability.

I. INTRODUCTION

Existing networks, such as the Internet, have focused on high data transfer rate and reliable data delivery. To achieve these goals, the network should perform lower header processing and should have effectively designed signaling. Over time, advances in hardware and transmission technology have allowed the user to support a high quality of data transmission.

Recently, reducing CAPEX/OPEX, deploying new network services, and orchestration and management are issues of growing importance in the network. In order to solve such problems, there are typical network paradigms such as Software Defined Networking (SDN), Network Function Virtualization (NFV), and autonomic networking [1][2][3].

To reduce OPEX/CAPEX and realize an autonomic network, we designed and implemented the Function Oriented Network (FON) that realizes a biology inspired autonomic network and can flexibly deploy new network services. Moreover, it enables the user and the service provider, as well as the network operator, to control the user plane of the network. The FON targets large scale Internet-of-Things (IoT) and access networks.

The rest of the paper is structured as follows. In Section II, we introduce the FON architecture and in Section III we describe its implementation. Section IV presents the conclusion and future work.

II. FUNCTION ORIENTED NETWORK ARCHITECTURE

In this study, we propose the FON architecture and SmartPackets to achieve network programmability by facilitating the rapid adoption of new network functions in network devices by network operators, service providers, and users.

In FON, a network node provides functions for network services on the basis of software and functions for network services, which are executed in terminals, where they are specified using open functions and transferred via SmartPackets. In addition, network services are managed by updating, distributing and removing functions for network services, which are executed in network devices via the network manager. Figure 1 provides an overview of the FON architecture and the structure of a SmartPacket. The FON comprises the FON device, FON node, and FON manager.

A. FON Device

The FON device is a database (DB) that stores function tables with function names and function codes, and a function processor, which creates SmartPackets and transfers them to a FON node, as well as receives SmartPackets from a FON node. The function processor receives function distribution messages from the FON manager, and stores and updates the function table information in the FON manager, including the message in the function table of the FON device. The function processor can perform SmartPacket verification using the function tables stored in the DB during SmartPacket generation. Through this procedure, a FON device can request and execute network functions in the FON node.

B. FON Node

A FON node is a DB that stores function tables, which contain executable function names, function code, and function usage counts (statistics). A function processor receives a SmartPacket and extracts information from the function call field to call the function that corresponds to the extracted function name, thereby performing the function. The function processor extracts the function code that needs to be executed from the DB and it performs dynamic binding to execute dynamic binding functions. In addition, the function processor can specify the function execution result in the SmartPacket payload and it transfers SmartPackets to other FON devices or FON nodes.

 ISBN: 978-1-61208-450-3

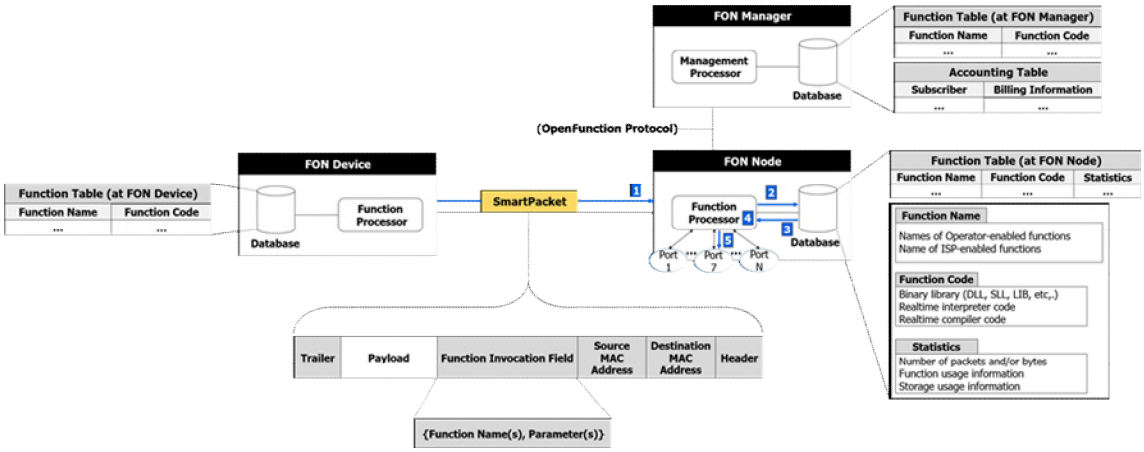


Figure 1. Function Oriented Network Architecture and SmartPacket Structure

C. *FON Manager*

The FON manager has a DB, which contains a function table with executable function names and function code, and an accounting table that manages billing information according to the function usage of the users and their subscriber information. The FON manager also has a management processor that extracts the function table from the DB and transfers it to the FON device or a FON node.

The management processor adds new functions that can be executed in a FON node to the function table of the FON manager and it transfers the function addition message, thereby adding the function name information and function code information to the function table of the FON node. The management processor transfers a function removal message to the FON node, thereby removing a specific function name information and function code information from the function table stored in the FON node. The management processor receives function usage information for each user from the FON node and it calculates billing information based on the function usage per user. It stores and manages this information in the accounting table.

D. *SmartPacket*

As shown in Figure 1, a SmartPacket is composed of the destination MAC header, the source MAC header, the function invocation field, the payload, and the trailer. The function invocation field includes the function name information that needs to be called, along with the required input parameters. It may contain multiple function names and input parameters. The output parameters for the execution results of functions that need to be called can be inserted into the payload.

III. IMPLEMENTATION

To implement the FON architecture, we built the testbed using a virtual environment, such as Virtual Box, and we implemented the FON Device and the FON Node using Linux virtual machine. The FON Manager was not implemented as part of this work and will be implemented in the future. The resources allocated for the virtual machine were Xeon E5520 2.27GHz CPU, 812MB RAM, 32GB SSD, and Gigabit Ethernet NICs. The FON Node and the FON Device were developed using the Python 2.7.9 programming language. The SmartPacket can be involved in many network function calls. Therefore, it is necessary to perform encoding and decoding of variable length message. The SmartPacket structure was implemented by using the pickle library of Python. Basically, FON should be implemented as an upper layer of MAC, but our prototype was implemented using TCP/IP protocol, such as socket programming. Ultimately, it will be implemented to L2/L3 based protocol.

IV. CONCLUSION AND FUTURE WORKS

In this paper, we introduced the FON architecture and described its implementation. The FON architecture can facilitate the rapid deployment of new network functions and protocols to satisfy requirements that are generated continuously. In future works, we will carry out performance evaluation and analysis through simulation and verification experiments. These experiments will target the processing overhead and message overhead, and will compare our approach with similar active network architectures [4]. After that, we will design and implement an ant-colony optimization based autonomic network using FON.

ACKNOWLEDGMENT

This work was supported by the ICT R&D program of MSIP/IITP, Republic of Korea. (B0101-15-1366, Development of Core Technology for Autonomous Network Control and Management), (B0190-15-2013, Development of Access Technology Agnostic Next-Generation Networking Technology for Wired-Wireless Converged Networks)

REFERENCES

[1] Naudts, Bram, et al. “Techno-economic analysis of software defined networking as architecture for the virtualization of a mobile network,” 2012 European Workshop on Software Defined Networking (EWSDN), Oct. 25, 2012, pp. 67-72.

[2] Hawilo, Hassan, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC),” IEEE Network, volume 28, issue 6, Nov. 24, 2014, pp. 18-26.

[3] M. Behringer, B. Carpenter, et al. “A Reference Model for Autonomic Networking,” draft-behringer-anima-reference-model-04, IETF, Oct. 16, 2015, pp. 1-24.

[4] David L. Tennenhouse and David J. Wetherall, “Towards an Active Network Architecture,” ACM SIGCOMM Computer Communication Review, volume 26, issue 2, Apr., 1996, pp. 5-17.

 ISBN: 978-1-61208-450-3

uLoBal: Enabling In-Network Load Balancing for Arbitrary Internet Services on SDN

Alex F R Trajano*, Marcial P Fernandez†
Universidade Estadual do Ceará
Fortaleza, Ceará, Brazil
Email: *alex.ferreira@uece.br, †marcial.fernandez@uece.br

***Abstract*—Today's networks should support the increasing demand required by large workloads generated by users who consume several types of service over the Internet. However, the users demand increases at a much higher rate than the networks can evolve due to a number of constraints, including economic reasons. Thus, it is a common practice to adopt load balancing in order to use network resources more efficiently. Although load balancers are an effective way of improving current networks, most of the existing implementations are based on hardware products and dedicated to specific types of services, which is not a good alternative due to the highly dynamic nature of the Internet. In order to address the nature of today's networks, the Software-Defined Networking (SDN) provides an architecture that allows the network to be fully programmable, opening the possibility of implementing such load balancing mechanisms on top of a network controller that provides optimal management of resources. This paper presents uLoBal, an SDN-based load balancer that is able of performing flexible and generic load balancing of arbitrary services through the use of manageable forwarding algorithms that address most of the service types and natures. uLoBal is efficient to load balance services on unstructured networks by considering both network and servers' load metrics. The proposal was evaluated in the Mininet emulation environment and shows an improvement on load balancing, providing better use of network resources and user experience.**

Keywords–Load Balancing; Software-Defined Networking; Internet Services.

I. INTRODUCTION

The growing complexity and workload of current computer networks often require large infrastructure investments in order to support new demands. However, it is not feasible to increase the network capacity at the same rate as the demand grows, requiring a set of techniques that aim at a more efficient use of network resources. One of the most-used techniques is to perform load balancing, either by application layer algorithms or network orchestration, in order to optimize network traffic.

In fact, over the last years, it has been common to find specialized hardware appliances or applications capable of performing traffic load balancing of specific types of service. However, the load balancing task should not be coupled with specialized infrastructure items, since it should be an embedded feature of the network itself. This approach, called in-network load balancing, is a way of providing more flexibility and near optimal performance, since it would not be needed to communicate with external devices for choosing to which network path a given packet should be forwarded. Besides, there is a wide set of Internet services that can be boosted using an in-network load balancing technique, which opens the opportunity for developing generic solutions focused on the adherence to current and future services at no cost.

The programmable network concept is coming back thanks to Software-Defined Network (SDN) architecture. SDN is an emerging approach that aims to provide a dynamic, manageable and adaptable platform that is ideal for the nature of current networks and applications. To do so, it decouples the network control from the data plane, enabling the network control to be fully programmable while the underlying infrastructure becomes specialized on data forwarding. The SDN architecture provides a networking environment that is centrally managed by a SDN controller, which is directly programmable, bringing more agility to business process due to the existence of a single point of control. Furthermore, the SDN architecture aims to be vendor-neutral and based on open standards, like the OpenFlow protocol, which is the main SDN technology at the moment.

This paper presents uLoBal, a SDN-based load balancer that is capable of performing flexible and generic load balancing of arbitrary services through the use of different forwarding approaches that address most types of service. uLoBal allows network administrators to manage which services are going to be load balanced and which algorithms will define how such process should occur. uLoBal supports the use of static and dynamic load balancing schemes, working on top of the SDN controller, helping to reduce the load on both the network and the server.

In the last few years, there has been some interest in implementing load balancing functions over SDN, but there has not been much interest in addressing multiple arbitrary services at a single point of management. While some works have focused on specific applications [1], others focused on particular network topologies [2] and others have scalability problems that may lead production networks to collapse [3]. uLoBal tries to gather the best characteristics of each previously proposed solution while providing a novel and efficient load balancing method.

uLoBal has been implemented and tested on the Mininet virtualized environment that emulates a real-world network [4]. The testing scenario was based on a Content Delivery Network (CDN), which is one of the most common types of service that is present on the current Internet.

This work is organized as follows. In Section II, we present some related works, while the basis of SDN and OpenFlow are shown in Section III. In Section IV, we present the uLoBal, the load balancing solution proposal. Sections V and VI show the experimental evaluation and the results. Section VII concludes the paper and presents some intended future works.

 ISBN: 978-1-61208-450-3

II. Related Work

Li et al. [2] have proposed an OpenFlow-based load balancer for Fat-Tree networks that supports multipath forwarding. Their proposal aims to recursively find the current best path from a source to a destination, load balancing the network by enabling the use of alternate paths at runtime, minimizing network congestion. Their algorithm works only on networks that operate on the Fat-Tree topology and use network metrics for choosing the best path.

Wang et al. [5] have proposed an interesting load balancing approach that aims to proactively load balance traffic from clients to servers by slicing the IP address space into trees that isolate a set of clients to a set of servers. The work uses the concept of server weighting, which assigns a fixed number of clients to a server on the network. To do so, the extensive use of wildcards is proposed, which may reduce forwarding performance and create management issues, as can be seen in [6]. Furthermore, the proposed solution requires that, under certain conditions (network topology changes or server weight updates), a part of the network traffic passes through the controller, what may lead to the collapse of the network controller. Network metrics are not considered.

Koerner et al. [3] proposed an architecture that enables in-network load balancing of multiple services using OpenFlow. Their proposal relies on a set of SDN controllers on top of a FlowVisor instance [7], where each controller is responsible for load balancing the traffic of a specific service. The authors have focused on the architecture, so there is no information about particular service implementation, while the performed experiment does not fit real-world scenarios. The idea of using a set of controllers to handle exact services might be interesting in some specific cases, but has the drawback of not permitting multiple services to be handled by a single controller, which is the most common case of SDN deployment.

Handigol et al. [1] show Plug-n-Serve, a module that resides within an OpenFlow controller that is capable of performing load balancing over unstructured networks, aiming to minimize average response time of HTTP servers. Plug-n-Serve load balances HTTP requests by gathering metrics about CPU consumption and network congestion on the network links, which enables its load balancing algorithm to select the appropriate server to direct requests to, while controlling the path taken by packets on the network.

III. SDN and OpenFlow overview

Software-Defined Network (SDN) is an approach to network control and management that allows administrators to manage network services by an abstraction of lower network protocol functions. This is done by decoupling the control plane, that takes decisions about forwarding traffic through the network infrastructure; from the data plane, the network traffic itself. The objective is to simplify the network control in high speed traffic. SDN requires a protocol for the control plane that is able to communicate with devices. The most known protocol is the OpenFlow, often misunderstood to be equivalent to SDN, but other protocols could also be used, such as Forwarding and Control Element Separation (ForCES) [8] and Network Configuration Protocol (NETCONF) [9]. In our work, we will consider only the OpenFlow architecture.

OpenFlow has two main components: the controller, an unique programmable remote control, and the network devices. These two components work together through the OpenFlow Protocol. The main idea is to keep network devices as simple as possible in order to reach better forwarding performance, having no complex decision-making process within the devices, delegating such a task to the network controller.

The OpenFlow Controller is the centralized controller of an OpenFlow network. It sets up all OpenFlow devices, maintains topology information, and monitors the overall status of the entire network. The OpenFlow Device is any OpenFlow-enabled device in a network, such as a switch, router or access point. Each device maintains a Flow Table that indicates the processing applied to any packet of a certain flow. The OpenFlow Protocol works as an interface between the controller and the switches setting up the Flow Table. The controller updates the Flow Table by adding and removing Flow Entries using the OpenFlow Protocol. The Flow Table is a database that contains Flow Entries associated with actions to command the switch to apply some actions on a certain flow. Some possible actions are: forward, drop and encapsulate. Figure 1 shows the structure of a Flow Entry.

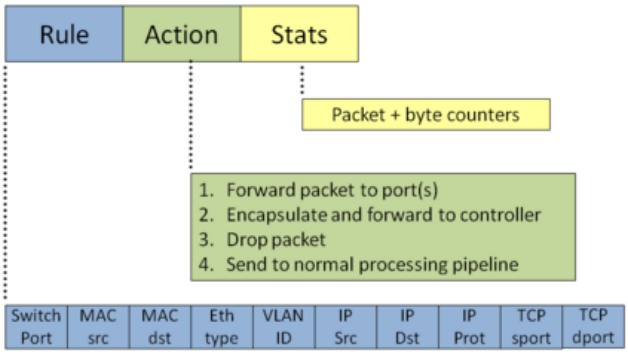


Figure 1. The OpenFlow Flow Entry [10].

Each OpenFlow device has a Flow Table with flow entries. A Flow Entry has three fields: Rule, Action and Stats. The Rule field is used to define the set of conditions to characterize the packets that will match that specific flow. The Action field defines the set of actions that must be applied to a packet if it matches the conditions defined in the Rule field. The Stats maintains a set of counters that are used to monitor flow's statistics, which can be used for management purposes. Each incoming packet is matched against the entries of the Flow Table. If a set of Flow Entries matches that packet, the device will select the entry with the highest priority, and the actions will be executed, having its statistics updated at the end of the process. If there is no matching entry for an incoming packet, the device sends a *PacketIn* message to the controller, wrapping the unmatched packet.

Once the controller receives a *PacketIn* message, it can take some actions on that packet, such as send *FlowMod* messages to the network devices in order to install new flow entries that match the incoming packet, or send a *PacketOut* message to "manually" forward the packet, or even ignore that packet. Besides, the controller can send messages to query statistics on every OpenFlow-enabled device within its network. An example is the *StatsRequest* message, which aims to query flow, port or queue statistics on the devices, which can be useful for determining the current state of the network. Each *StatsRequest* message is sent asynchronously, so the controller

 ISBN: 978-1-61208-450-3

must listen for a *StatsResponse* message that will gather the requested statistics' data.

IV. uLoBal: Enabling Load Balancing on SDN

The uLoBal architecture was designed to provide high flexibility and to fit multiple and diverse scenarios and needs. As described previously, it is essential that an in-network load balancer can address different types of service, not being focused on specific scenarios. The uLoBal follows such design directive and aims to identify a set of servers of a service using a unique identifier that will set the load balancing mode for that service. Furthermore, the uLoBal needs to be aware of network statistics in order to enable a load balanced forwarding, allowing a more efficient use of network resources.

To this extent, uLoBal has three main modules: (1) the network monitoring module; (2) the load balancing module; (3) the management module, a Representational State Transfer (REST) Application Programming Interface (API) that allows management by the network administrator and integration with other monitoring tools. All these components are embedded on the SDN controller in order to avoid unnecessary communication with external services. Figure 2 shows how these components are connected.

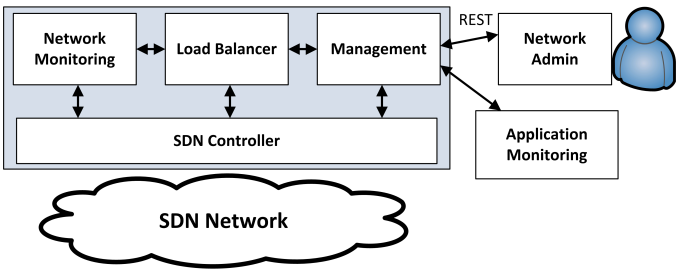


Figure 2. uLoBal architectural modules integration.

In an environment where there is a set of services that can be accessed through multiple endpoints, it is possible to perform in-network load balance in order to allow an even distribution of requests. The main objective behind uLoBal is to enable such service load balancing yet performing network load balancing. In networks where there are multiple paths between clients and endpoints, it is possible to use alternate paths as the network workload grows, mitigating problems related to networking congestion and reducing end-to-end latency. uLoBal can load balance the servers of the services by algorithms that use static and dynamic approaches that account for recent load information on both servers and network. Further subsections will give detailed information about each module.

A. Management Module

The uLoBal Management API can be accessed through a REST service, in order to allow administration and integration with external systems that can give updated load information of the services' servers. As the SDN controller cannot perform complex load monitoring at the servers, it is necessary to expose such service in order to allow an external monitoring tool to provide such information to the load balancer. Table I shows the uLoBal API methods.

The uLoBal uses a tuple of three values to identify an endpoint (or server) of a service: the *ServiceId*, *IP* address and transport *port* values. The *ServiceId* value is any string that uniquely identifies the provided service on a set of servers, being each server identified by the *IP* address and transport *port* values. Any kind of service that uses the TCP or UDP protocols can be addressed using these values if all of its endpoints are accessed on the same transport port, which is the most common case. Methods 1 and 2 of the API use exactly this tuple in order to add or remove endpoints to/from the load balancing. Method 3 is used by an external monitoring system that updates the load information of each server that provides access to the service, being the *Load* value the last load measure of a server normalized within the interval $[0, 10]$. The *Load* is a generic value that can be calculated using any application's specific metric. In order to allow the network administrator to change the load balancing mode that must be used for a given *ServiceId*, the *Mode* value must be provided to method 4 using one of three possible values: *ServerRoundRobin*, *ServerIpHash* or *NetServerLoad*, making the load balancer to change the balancing algorithm.

TABLE I. uLoBal API methods and parameters

#	method	parameters	used by
1	insertServiceEndpoint	(ServiceId, IP, Port)	Network Admin
2	deleteServiceEndpoint	(ServiceId, IP, Port)	Network Admin
3	updateServerLoad	(ServiceId, IP, Port, Load)	Monitoring System
4	changeLBMode	(ServiceId, Mode)	Network Admin

B. Network Monitoring Module

Since uLoBal uses network load information in order to load balance the service traffic on multiple forwarding paths, it is necessary to account for such load data to perform the load balancing. The network monitoring consists of two steps: (1) collect statistics on every port of every switch of the SDN at predefined time intervals; (2) calculate the minimum spanning tree of the network graph using the collected statistics as the cost metric.

The collecting process is made through the use of *StatsRequest* messages sent from the SDN controller to all switches on the network. Adrichem et al. describe a similar process in [11]. When the load statistics are collected, the Dijkstra algorithm is used to compute the minimum spanning tree that will be internally cached to be queried by the load balancing component. The cost metric used to compute the tree is given by $LinkCost = b + e$, where b is the percentage of the used link's bandwidth and e is the percentage of packets that have suffered of either drops or transmission errors. The $LinkCost$ must be normalized within the interval $[0, 10]$ before the spanning tree is calculated.

C. Load Balancer Module

The uLoBal load balancer module is responsible for load balancing requests based on three operational modes: Round-Robin (RR), IP Hashing (IPH), and Network and Server Load (NSL). Each one is identified in the REST API by *ServerRoundRobin*, *ServerIpHash* and *NetServerLoad*, respectively.

The load balancing mechanism is based on the principle of SDN, where the controller can push flows on the switches when there is no matching flows for an arriving packet. At this moment the controller receives a *PacketIn* message, that will be handled by the load balancer module if the destination IP and

 ISBN: 978-1-61208-450-3

port match some previously inserted endpoint. The handling algorithm will depend on the configured load balancing mode for the matching ServiceId, with the NSL mode as the default mode. Algorithm 1 shows how the *PacketIn* message is handled by the controller.

Algorithm 1: PacketIn handling algorithm

```
Data: A PacketIn message
if Packet's destination IP and port belongs to any ServiceId then
    sId := get the matching ServiceId; lbMode := get the
    configured load balancing mode for sId;
    switch lbMode do
        case ServerRoundRobin
            call RR(PacketIn, sId);
        end
        case ServerIpHash
            call IPH(PacketIn, sId);
        end
        case NetServerLoad
            call NSL(PacketIn, sId);
        end
    endsw
    Send the packet on a PacketOut message;
end
else
    Ignore the packet, not interfering the normal processing;
end
```

The uLoBal provides two approaches for load balancing service requests. The first is static, an approach that does not consider either network or server metrics in order to make decisions to where forward incoming traffic, while the second approach is dynamic and uses this metrics in order to make traffic orchestration. The static approach has the advantage of less overhead since there is no need to be aware of such metrics, which may be useful for networks where there is little congestion and to services that need some predictability about which server will handle a given request. On the other hand, the dynamic approach enables better network traffic orchestration, using resources according to their most up-to-date metric information, which can help to reduce network congestion and, consequently, improve the users Quality of Experience (QoE). Algorithms 2 and 3 use the static approach, while the Algorithm 4 uses the dynamic approach.

Algorithm 2: RR algorithm

```
Data: A PacketIn message and the sId
Increment the RR packet counter for the given sId;
pktC := the RR packet counter for the given sId;
sLen := get the amount of servers that belongs to sId;
sIndex := pktC mod sLen;
From the list of servers of sId, get the server dstSrv stored at
the sIndex position;
Get the less costly network path from the packet's source to
dstSrv and send FlowMod messages to the switches on the
path;
```

Algorithm 2 is a basic function that simply forwards requests by choosing the destination server through the Round-Robin algorithm. Once it chooses the server, it gets the cheapest current network path from the source to the destination in order to load balance the network. Its main characteristic is that the servers constantly receive a similar amount of requests, which can be useful for services that the costs of the requests are always the same.

Algorithm 3: IPH algorithm

```
Data: A PacketIn message and the sId
Create a circular list srvCirLst;
foreach server srv that belongs to sId do
    Calculate the hash h of the srv IP;
    Insert h into srvCirLst;
end
Get the packet's source IP and calculate its hash srcH;
Insert srcH into srvCirLst and get its index srcIdx;
Get the server dstSrv whose hash is stored at the srcIdx + 1
position on srvCirLst;
Get the less costly network path from the packet's source to
dstSrv and send FlowMod messages to the switches on the
path;
```

Algorithm 3 aims to map a set of clients to the same endpoint following a Consistent Hashing approach [12]. The main goal is to always forward requests made by a user to the same endpoint, which may be useful for services that need to fetch context information before serving the request, since this context information can be locally cached.

Algorithm 4: NSL algorithm

```
Data: A PacketIn message and the sId
Create a Hash Map cstMap capable of storing multiple values
mapped by a single key;
foreach server srv that belongs to sId do
    Get the less costly network path nPth from the packet's
    source to srv;
    netCst := the cost of nPth;
    srvCst := the current srv cost value;
    cost := ∛(netCst × srvCst);
    Insert the nPth on cstMap mapped by cost;
end
Get the minimum key k from cstMap;
Get a random entry nPth mapped by k;
Send FlowMod messages to the switches on the path nPth;
```

Algorithm 4 was designed for considering the current network and server loads in order to choose the destination server. It works by selecting the endpoint that can be accessed with the minimum cost, being the cost calculated through the geometric mean of both server and network costs. As a dynamic approach, it is not easy to predict which requests are going to reach a determined server, since it will depend exclusively on the current load from both servers and network.

Note that Algorithms 2, 3 and 4 send *FlowMod* messages to the switches within the network path from the source to the selected endpoint. Each *FlowMod* message consists of a header that matches the packet and two actions, (1) rewrite the packet's destination/source address; and (2) send the packet to the next hop. Furthermore, note that even though Algorithms 2 and 3 perform a static server load balancing, the chosen network path remains dynamic, since the selection process is based on the network metrics collected by the monitoring module. The flows generated by uLoBal use a *soft timeout* approach in order to set the duration of flows on switches, configuring the inactivity timeout to 5 seconds.

V. Evaluation

In order to evaluate how the load balancer would behave in production networks, two scenarios were elaborated to evaluate the effectiveness of the proposal and give some insights about further deployment.

 ISBN: 978-1-61208-450-3

The first scenario aims to compare how the network load balancing approach affects the clients perceived delay when requesting some content on a CDN server that operates through HTTP. This scenario aims to evaluate only the network load balancing by avoiding the server load balancing, allowing better analysis of the proposal behavior. To this extent, the network topology has only a single server that is accessed by several clients spread over the network. Since there is a single server, all the requests are forwarded to a single point of the network, making the load balancer change the forwarding paths at runtime, balancing the traffic load. For comparability, the experiment has addressed the use of the proposed load balancer operating on the NSL mode and the use of a traditional Shortest Path First (SPF) approach.

The second scenario aims to compare how the different load balancing modes affect (1) the client perceived latency and (2) the server load. Again, the clients are going to request contents on CDN servers that operate through HTTP. In this scenario, we used three CDN servers, each one providing an endpoint for content delivery, making the load balancer forward requests to one of these servers, following the configured load balancing mode.

The evaluation environment was based on a virtualized network using Mininet [4]. The Mininet system permits the specification of a network interconnecting virtualized devices. Each network device, hosts, switch and controller are virtualized and communicate via Mininet. A Python script is used to create the topology in Mininet, and the traffic flow control is made by the OpenFlow controller. Therefore, the test environment implements and performs the actual protocol stacks that communicate with each other virtually. The Mininet environment allows the execution of real protocols in a virtual network. The possibility to set link bandwidth and delay in Mininet allowed us to perform an experiment similar to an actual real scenario. The chosen OpenFlow controller was the Floodlight [13], due to its simplicity and development flexibility.

Figure 3. Network Topology. http://c-bgp.sourceforge.net/tutorial.php

Figure 3 shows the Abilene network topology, which has been used to perform the experiments on top of Mininet. The topology's sources were obtained from the TopologyZoo [14] and parsed according to the method described by [15]. In the first scenario, the server was positioned at the network node represented by the Indianapolis city. In the second scenario, the servers were positioned at the New York, Washington and Sunnyvale cities. Neither link latency nor bandwidth has been modified in the experiments.

In each city that did not contain any CDN endpoint, a set of 10 clients has been placed in order to make content requests. Each client was configured to perform sequential requests to a randomly chosen server from the available servers of the experiment. Of course, it is the job of the load balancer to redirect the request to the proper server, following the load balancing mode.

VI. Results

After each client finished 10^6 requests, the experiments results were collected and the graphs in Figures 4, 5 and 6 were built.

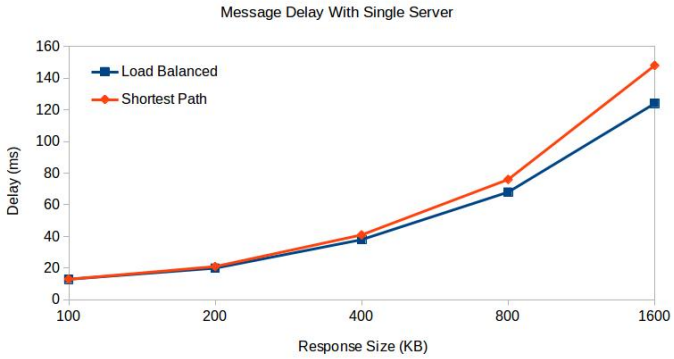


Figure 4. Results of the first experiment.

Figure 4 shows the results for the first experiment, where the network load balancing was compared to a SPF approach. It is possible that at low workloads, these approaches did not show significant differences, suggesting that the proposed network load balancing scheme is not relevant. However, when the workload starts to grow, there is an improvement in order of tens of milliseconds on the clients perceived delay, suggesting that this network load balancing approach can be useful as the workload grows. Besides, it is possible to conclude that at high workloads, the network would benefit form such load balancing mechanism, since congested paths would be avoided.

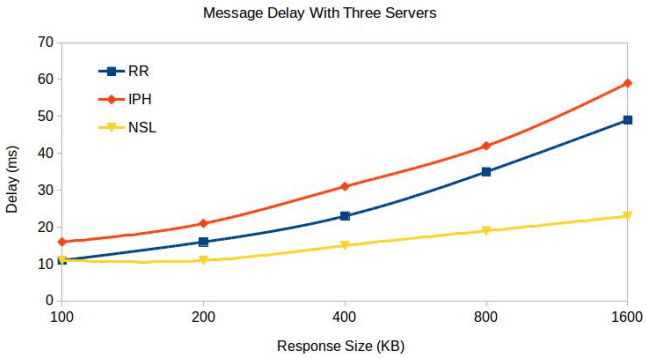


Figure 5. Results of the second experiment, from the point of view of the clients perceived delay.

Figure 5 shows the results for the second experiment, where the load balancing modes can be compared with each other. Both RR and IPH have similar behaviors at different workloads, although the RR shows better latency results. Such results can be explained by the fact that the network nodes are spread over an area of a whole country, with the servers being positioned at the edges of the network, which makes the use of these techniques a bad idea due to the high distance, increasing the end-to-end delay. It is possible to notice that the NSL mode outperforms both RR and IPH modes, which can be explained by the use of both server and network metrics when deciding to which server the requests will be forwarded, considering always the less costly network path at the moment. When the responses' sizes were 1600 KB, the improvement was about 53% and 62%, compared to RR and IPH modes, respectively.

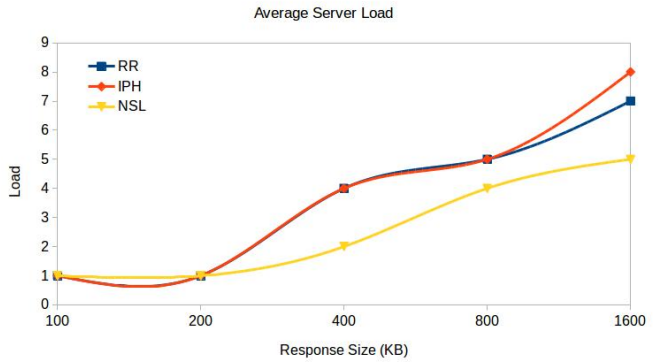


Figure 6. Results of the second experiment, from the point of view of the servers average load.

Figure 6 shows the results for the second experiment, where the average servers' load can be compared according to the load balancing mode. Again, both RR and IPH modes are similar to different workloads, unless the responses' sizes were 1600 KB. This result was expected, since these load balancing schemas aim to distribute the requests evenly across the servers, not looking for any external factor on the decision-making process. For this reason, it is also possible to notice that NSL can alleviate the average server load in most of the workloads, suggesting that the proposed load balancer can be useful in different production networks with distinct workloads when configured to operate in this mode. Furthermore, it was not observed any significant change in the consumption of resources in the network controller, even when using the NSL mode, suggesting that uLoBal follows the same scalability levels of the network controller.

VII. Conclusion and Future Work

This paper has presented uLoBal, a SDN-based load balancer that is capable to load balance arbitrary services through the use of different forwarding approaches that address services of several types and nature. The work showed that uLoBal works by aggregating a set of servers that defines the services' endpoints, allowing the network administrator to enable the load balancing of requests through the use of three different load balancing modes. Besides, uLoBal supports the use of static and dynamic load balancing, making it adaptable to different types of service. Experimental results showed that the network load balancing performs better when compared to classic network forwarding, while enabling load balancing at the servers of distributed services.

As a future work we intend to aggregate more load balancing modes on the current implementation of uLoBal, while optimizing the existing algorithms to provide even better performance. Since uLoBal can work over unstructured networks, it is needed to verify how the network topology can affect the performance of each load balancing mode, which can give important insights about the positioning of servers on the network according to the type of service.

References

[1] N. Handigol, S. Seetharaman, M. Flajslik, N. McKeown, and R. Johari, "Plug-n-serve: Load-balancing web traffic using openflow," ACM SIGCOMM Demo, vol. 4, no. 5, 2009, p. 6.

[2] Y. Li and D. Pan, "Openflow based load balancing for fat-tree networks with multipath support," in Proc. 12th IEEE International Conference on Communications (ICC'13), Budapest, Hungary, 2013, pp. 1–5.

[3] M. Koerner and O. Kao, "Multiple service load-balancing with openflow," in IEEE 13th International Conference on High Performance Switching and Routing (HPSR2012). IEEE, 2012, pp. 210–214.

[4] B. Lantz and B. Heller, "Mininet: rapid prototyping for Software Defined Networks," Last accessed, Aug 2015. [Online]. Available: http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Mininet

[5] R. Wang, D. Butnariu, J. Rexford et al., "Openflow-based server load balancing gone wild," in Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 2011, 2011.

[6] B. Lopes Alcantara Batista, G. Lima de Campos, and M. Fernandez, "Flow-based conflict detection in openflow networks using first-order logic," in Computers and Communication (ISCC), 2014 IEEE Symposium on, June 2014, pp. 1–6.

[7] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," OpenFlow Switch Consortium, Tech. Rep, 2009.

[8] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," RFC 5810 (Proposed Standard), Internet Engineering Task Force, Mar. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5810.txt

[9] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6241.txt

[10] Open Networking Foundation, "Openflow switch specification, version 1.3.5," Last accessed, Sep 2015. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf

[11] N. L. Van Adrichem, C. Doerr, F. Kuipers et al., "Opennetmon: Network monitoring in openflow software-defined networks," in Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014, pp. 1–8.

[12] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. ACM, 1997, pp. 654–663.

[13] D. Erickson, "Floodlight Java based OpenFlow Controller," Last accessed, Aug 2015. [Online]. Available: http://floodlight.openflowhub.org/

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," Selected Areas in Communications, IEEE Journal on, vol. 29, no. 9, 2011, pp. 1765–1775.

[15] M. Großmann and S. J. Schuberth, "Auto-Mininet: Assessing the Internet Topology Zoo in a Software-Defined Network Emulator," Otto-Friedrich University, Tech. Rep., 2013.

 ISBN: 978-1-61208-450-3

Smart Factory or Smart Car?

The concept of IoT usability

Kazuyuki Shimizu
School of Business Administration
Meiji University
Tokyo, Japan
shimizuk@meiji.ac.jp

Abstract— **The subject of this paper focuses mainly on the world car industry and, especially, on how to implement Internet of Things (IoT) technology into the Japanese, German, and U.S. car industries, as well as on differences between the concepts of IoT usability in those target countries. The car industry is now at a difficult phase with regard to how to adjust secret information for car production using open-access IoT technology. Intercorporate relationships provide advantages for car production in general; however, this open-access IoT technology is likely to completely re-arrange these relationships. It will be very interesting to see how these relationships will re-arrange. In this paper, we apply the "smiling curve concept" to find a solution for differences in IoT usability in the target countries. Then, we assess producers' ideas, such as the smart factory in Germany and Japan, and the users' need for a smart car in the U.S.A. As a result, we explain the advantages of intercorporate relationships and how integral parts of the product will be replaced using IoT technology.**

Keywords-component; Internet of Things (IoT); INDUSTRIE 4.0; Toyota; smiling curve; cyber and physical systems (CPS).

I. INTRODUCTION

German car and manufacturing industries are trying to connect their production equipment through the IoT to reduce their inventories, costs and time. These attempts represent the early stages of smart factory, which requires an advanced Supply Chain Management (SCM). Germany is currently ranked third in sales of manufacturing equipment in the world, according to data from Gardner Research in 2015 [1]. This trend is dependent upon the number of cars in production, because car components are made by machines from the manufacturing industries. The existing car industries could shape the former "Lean Production" idea, to reduce inventories [2].

In our work, we conducted a comparative study of producer's and user's viewpoints for car industries. German Volkswagen (VW) and Japanese Toyota are focusing on their production systems. Their production systems are different: VW has a module based production system (Modularer Querbaukasten; MQB) and Toyota has a lean based production system (Toyota New Global Architecture; TNGA). We will not present the details of the difference between the U.S. and the above mentioned European module based production system. The two systems are hypothetically treated as being the same for the purpose of this paper.

Both Toyota and VW have different ways of using the IoT technology. We believe that the parts combination influences the entire idea of IoT usability, which is causing differences between producer's and user's viewpoints. U.S. companies are run by user's viewpoint, for example, through wearable technology such as WiFi, Bluetooth or other connective standards. German and Japanese companies are not run by user's viewpoints, but, rather, VW and Toyota think about the producer's viewpoints, which are much more related to manufacturing special parts combination and the smart factory.

The rest of the paper is structured as follows. In Section 2, we address the question of what a "smart factory" is in Germany, with the INDUSTRIE 4.0 program. In Section 3, we describe the car parts segmentation and smart, connected products that use a similar architecture to that of computer segmentation. Section 4 presents the differences between Volkswagen and Toyota and Section 5 explains INDUSTRIE 4.0 in the Japanese context. The "smiling curve" and the interaction between the cyber and physical space and addressed in Sections 6 and 7, respectively. We conclude the paper in Section 8.

II. INDUSTRIE 4.0 (SMART FACTORIES)

German car and manufacturing industries are trying to use IoT technology to enhance their competitiveness. The name "INDUSTRIE 4.0" means the upgrade of industrial revolution from 1.0 to 4.0. The first industrial revolution '1.0' followed the introduction of water- and steam-powered mechanical manufacturing facilities in the 17th century. The second industrial revolution '2.0' followed the introduction of electrically powered mass production based on the division of labor between the 18th and the 19th century. The assembly line was introduced on a large scale by the meat-packing factory in Chicago. The third industrial revolution '3.0' used electronics and Information Technology (IT) to achieve further automation of manufacturing in the 20th century. Now, companies are beginning to interactively connect their machinery equipment through the IoT, to improve the industrial processes involved in manufacturing, engineering, material usage, and supply chain as well as life cycle management. This is called the industrial revolution '4.0', which is based on using Cyber-Physical Systems (CPS) by an establishing global networks and common sensors [3].

 ISBN: 978-1-61208-450-3

CPS make the link between computational Cyber and Physical elements supported by widely used cheap sensors. Recently, the automotive industry is characterized by a static production line. However, German 'INDUSTRIE 4.0' could offer a dynamic production line. Each parts components "Module" will be linked by the sensors mentioned above. This will be a result of producer's viewpoints.

On the other hand, Google made a self-driving vehicle (Physical space). This product uses a CPS scheme. There are different types of sensors such as GPS (Global Positioning System) receiver, Laser range finder, Video camera and Radar. These types of common sensors bring in big data into Cyber space. There is also an interaction between big data in Cyber space and the Physical system. This is why we think U.S. car manufacturing is much more dependent on user's viewpoints.

There is high level IoT planning around the world, such as "Advanced Manufacturing National Program (AMNP)" [4] in U.S.A., "La nouvelle France industrielle" in France, and "Future of Manufacturing" in the U.K. The German "INDUSTRIE 4.0" is also one of these high-level national IoT plans [5]. This high level planning creates a competition between several countries in terms of their technological innovative viewpoint.

III. CAR PART SEGMENTATION AND SMART, CONNECTED PRODUCTS

A car is made up of more than 30,000 parts that can be divided into five broad categories, namely: Engine, Drivetrain, Body, Chassis and Internal/Body Interior.

- The engine is the heart of the car, and it converts thermal energy into mechanical energy.
- The drivetrain transmits the output of the engine to the wheels, via the transmission, drive shaft and differential.
- In general, the body refers to the bonnet, doors, trunk of the car and essentially consists of steel.
- The chassis is currently defined as including the suspension, steering, tires, and wheels.
- The body interior is also an important component of the car, and includes the seats, dashboard, air conditioner, and audio

Cars may be equipped with additional items to enhance their comfort.

Table I shows the different car parts segmentation. VW and Toyota have developed highly innovative production systems that give modularity considerable thought. The term modularity is widely used in studies of technological and organizational systems. Product systems are deemed "modular" when they can be deconstructed into a number of components that can be mixed and matched in a variety of configurations. For example, an internal body "Interior" (each of dash board, air conditioner and audio) forms one packaged "module". This "module" will be linked with a centralized main computer support by widely used cheap sensors.

TABLE I. CAR PART SEGMENTATION

Car Part Segmentation	I. Engine	II. Drivetrain	III. Body	IV. Chassis	V. Interior
Toyota	***Hybrid Electricity***	Drive Wheels	Bonnet	Suspension	Sheets
	Hydrogen Engine	Transmission	Doors	Steering	*Dash boards*
Volkswagen	***Down Sized Engine***	Drive Shaft	Trunk	Tires	*Air Conditioner*
	Ecofuel Engine	Differential		Wheels	*Audio*

A computer is actually one of the best examples of modular design. The typical modules for this design are power supply units, processors, mainboards, graphics cards, hard drives, optical drives, etc. All of these parts could be easily changeable, so if the parts are used by standard interface, they can be easily replaced.

M. Porter and J. Heppelmann [6] talk about the historical transition of IT into the product value chain. During the 1960s and 1970s, with the first wave of IT, automated individual activities in the value chain used a computer. For example, order processing and bill paying was done through a computer. In the 1980s and 1990s, the second wave of IT-driven transformation included an inexpensive and ubiquitous connectivity to the Internet. The two waves gave rise to huge productivity gains and growth over the economy. However, the products themselves were largely not affected. Then, in the third wave, IT is becoming an integral part of the product itself, with embedded sensors, processors, software, and connectivity in products, coupled with a product cloud in which product data is stored and analyzed. There are called "Smart, Connected Products".

IV. VOLKSWAGEN (MQB) AND TOYOTA (TNGA)

Regarding the concept of "modular design" used in the production system at German VW and Japanese Toyota, we compared, for example, their engine varieties. Table I shows the variety of "I. Engine" module. Here, we notice that Toyota has an advantage of Hybrid engine on one hand, but VW has a competitive edge because of a down-sized engine, on the other hand. This competitive advantage is a product of their inter-corporate relationships. Japanese car producers are characteristically vertically integrated, which is different from an Anglo (U.S. and Europeans) horizontally integrated model. Japanese car producers need, for example, monthly and weekly meetings, which include a variety of other networks such as banks, trading houses, suppliers and customers, which is called "Keiretsu". On the other hand, an Anglo (U.S. and Europeans) model is horizontal integrated. These inter-corporate relationships influence their parts combination for MQB and TNGA.

VW has developed a very innovative parts combination called the Modular Transversal Toolkit (MQB) with their suppliers. They went through a tough period with their production system in the 1980s. MQB was developed under this tough environment. German workplaces are very strictly protected by trade unions, and the existing employee structure created a barrier to change their production system.

 ISBN: 978-1-61208-450-3

We previously wrote a paper about "German Heavy-weight Management: A Case Study of Volkswagen", which described the potential competitiveness of well-developed German production systems [7].

The MQB system has the engine "quer," or transverse. The "Modularer Längsbaukcombinationasten" (MLB) and "MMB" (Modularer Mittelbaukasten) systems are applied to bigger cars. This development of each broadly formed module such as from MQB, MLB to MMB aims for a more standardization of parts, which is enhanced from the economical and mass markets products to the a middle luxury class to reduce the cost and assemblies.

Basically, Toyota has a long time value chain, which is a tightly coupled production system. The client's diversification led to environmental changes by Toyota New Global Architecture (TNGA). This new platform realizes optimal design freedom and ergonomics. The TNGA provides a foundation for grouping development, which enables the standardization of parts and components across different models, such as VW's parts combination developments, improving the efficiency of the development process while reducing costs.

V. INDUSTRIE 4.0 (IOT) IN THE JAPANESE CONTEXT

Another question is how to find the best matching point between the standardized (MQB) and elaborated components (TNGA) for each type of integration. U.S. and Germany have created several types of consortium, such as internal combustion, robotics and others etc. U.S. and Germany are very well organized horizontally in terms of their knowledge cooperation among government, industry and academia, not like Japanese vertical intercorporate relationship. This is both good and bad at the same time. Wiener says, "These advantages would seem to be the ability of the brain to handle vague ideas, as yet imperfectly defined." And "Render unto man the things which are man's and unto the computer the things which are the computer's [8]."

VI. SMILING CURVE

The founder of Aser.co, Stan Shih [9] argued the concept of the "Smiling Curve" in 1992. This theory is about appropriate value added of different sections in the industry. Figure 1 below shows the "smiling curve".

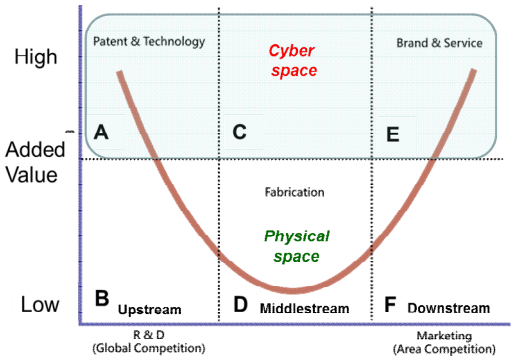


Figure 1. Smiling Curve.

A smiling curve is an illustration of value-adding potentials of different components of the value chain in an IT-related manufacturing industry.

The concept of "Competitive Advantage" written by M. Porter in 1985, gave the concept of the "value chain" [10]. The primary activities in the value chain are: 1. Inbound Logistics 3. Operations 4. Outbound Logistics 5. Marketing & Sales and 6. Service. Also, support activities include: A. Firm Infrastructure B. Human Resource Management C. Technology Development, and D. Procurement. Porter suggests that a firm might develop a competitive advantage in any one of these areas; 1 to 6 and A to D.

The value chain command higher values added to the product (such as Patent & Technology, R&D and Brands & Service) than the middle part (such as Production) of the value chain. Also the Fabrication part could be divided between many different shops, such as paint, assembly, body and so on.

The basic structure of the "Smiling Curve" in Figure 1 from left to right on the horizontal axis, is the upper, middle and downstream of an industry, that is, the component production, product assembly and distribution. The vertical axis represents the level of value-added. In terms of market competition type, the left side of the curve is worldwide competition whose success depends on technology, manufacturing and economy of scales. On the right side of the curve is regional competition. Its success depends on the brand name, marketing channel and logistics capability.

In Figure 1, the "Smiling Curve" explains, from left to right on the horizontal axis, which is labeled A, C and E on this diagram called Cyber space. In this area, we could add a high value virtually, such a patent and idea of technology. This sphere contains intellectual properties, such as Software.

The bottom part of the conceptual diagram, labeled B, D and F, shows what is called the "Physical space" . This area gives a limitation for the added value to physical existence, which is a real product. A good example is assembled car products, in this context.

VII. INTERACTION BETWEEN CYBER AND PHISICAL SPACE

In this paper, we applied the concept of "smiling curve" to car manufacturing industry. The way the automotive industry uses modular production in today's global competitive environment is similar to the way computers are produced. IoT usability is different between Japanese and Anglo (U.S. and Europeans) countries in terms of the car industry. We discussed about two different types of intercorporate relationships, which are the Japanese ("Keiretsu"= vertically integrated system) and Germans-U.S. type (horizontally integrated system).

Figure 2 positions the targeted countries U.S., Germany and Japan on the "Smiling Curve". The figure shows the Japanese "Keiretsu"= vertically integrated system and Anglo (U.S. and Europeans) type horizontally integrated system, as well as IoT usability in producer's and user's viewpoints.

Assembled car products are mainly made from German and Japanese car producers. Therefore, we think that the

 ISBN: 978-1-61208-450-3

German and U.S. viewpoint for the car industry are different in terms of using the IoT technology. Japanese and German car producers are focusing more on the physical way for assembly production line out of their module (parts combination) into the real world, using the IoT technology. This plays a very important role for their supply chain management (SCM) such as "Keiretsu" and Anglo horizontally integrated system [11].

VIII. CONCLUSION

In conclusion, in this paper, we emphasize the different concepts of IoT usability in targeted countries such as Japan, Germany and U.S.A. German and Japanese car producers are more focused on factory management using IoT technology. Germans as well as U.S. place considerable emphasis on user's viewpoint, which means using a car more as an equipment. The car is comparable to a computer, collecting information through IoT (common sensor) technology. Then, we see how the highly developed U.S. user's viewpoint could combine with Japanese and German producer's viewpoint to find out innovative products for smart factories, which is now required.

Google is now trying to make a complete self-driving vehicle without the necessary knowledge of car production.

They have much more IoT based knowledge for car production. Japanese and Germans are adopting this self-driving technology in a stepwise manner, introducing it into their production line, which is needed for the interaction between Cyber and Physical space.

We conducted a module based production system MQB of VW and the other lean based TNGA. The three largest U.S car manufacturers develop their advantage on the production method by using the Smart Factory (big-data) idea. "Smart" means to use "big data", and use it in an intelligent manner.

The meaning of a competitive advantage has been changed with this newly developed relationship with regard to the whole transport environment system.

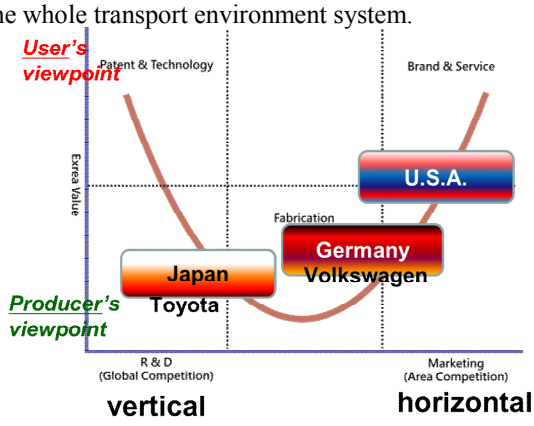


Figure 2. Positions of Japan, Germany and USA on the Smiling Curve

Japanese and German manufactures get their competitive advantage from their intercorporate relationships, not from drivers as user's point of view. However, Japanese vertical ("Keiretsu") model and Anglo horizontal model are substantially developed. As we've discussed by the "Smiling Curve" and CPS, one could make the link between computational Cyber and Physical elements supported by a wide use of cheap sensors. A car becomes a sensor and could re-arrange the total transportation environment.

"Keiretsu" vertically integrated system has produced TNGA. Germans-U.S. horizontally integrated system has developed MQB. Therefore, there is a lot of high level IoT planning around the world. In the car industry, both user's and producer's viewpoints have advantages and disadvantages. However, a horizontally integrated system is far more advanced than TNGA in terms of cyber space connectivity. Modules are developed following the computer based architecture. Such integral parts of the product will be replaced using IoT technology.

REFERENCES

[1] Gardner Research, "World Machine-Tool Output & Consumption Survey 2015", p4, 2015.

[2] K. Clark and T. Fujimoto, "Product Development Performance", Japanease translation rights arrenged with Harvard Business School Press in Boston. through The Asano Agency, Inc. in Tokyo. pp321-338, 2001.

[3] H. Kagermann, W. Wahlster and J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", in resaerch paper from "Forschungsunion" and "acatech", pp17-21, 2013.

[4] Executive Office of the President President's Council of Advisors on Science and Technology, "Report to the President Accelerating U.S. Advanced Manufacturing", p4, Oct 2014.

[5] C. Siemieniuch, M. Sinclair and M. Henshaw, "Global drivers, sustainable manufacturing and systems ergomics", Applied Ergonomics, No.51, pp104-119, Elsevier, 2015.

[6] M. Porter, and J. Heppelmann, "How Smart, Connected Products Are Transforming Competition", Harvard Business Review, pp64-88, Nov 2014.

[7] K. Shimizu, "German Heavy-weight Management: A Case Study of Volkswagen", in Proceeding of the APCIM 2012, August 2012.

[8] N. Wiener, "God and Golem, Inc., A Comment on Certain Point where Cybernetics Impinges on Religion", The M.I.T. Press, Cambridge, pp72-73, 1964.

[9] S. Stan, "Growing Global; A Corporation Vision Masterclass", John Wiley & Sons (Asia) Pte Ltd, pp10-11, 2002.

[10] R., Ankli, "Michael Porter's Competitive Advantage and Business History", Business and Economic History", Second Series, Vol.21, pp231-233, 1992.

[11] S. Turnbull, "The competitive advantages of stakeholder mutuals", 12th Annual Meeting on Socio-Economics Society for the Advancement of Socio-Economics, LSE, p8, Last revised September 20th, 2000.

 ISBN: 978-1-61208-450-3

A Study on Device Management for IoT Services with Uncoordinated Device Operating History

Megumi Shibuya[†], Teruyuki Hasegawa[†] and Hirozumi Yamaguchi[‡]
[†]KDDI R&D Laboratories, Inc.
Saitama, JAPAN
e-mail: {shibuya, teru}@kddilabs.jp
[‡]Osaka University
Osaka, JAPAN
e-mail: h-yamagu@ist.osaka-u.ac.jp

***Abstract*—The cumulative failure rate is an important reliability index for evaluating Internet of Thing (IoT) device and system quality and reliability. However, in the horizontal specialization business model, IoT service infrastructure is often operated by multiple players such as service providers and device vendors, and device management information necessary to obtain the cumulative failure rate are owned independently and uncoordinatedly by them. In this paper, we propose a method of calculating the cumulative failure rate in such environment. We design an algorithm to aggregate and organize such distributed, uncoordinated information to derive the device operating history, which is fed into the cumulative failure rate calculation formula. Through several simulation experiments, we show the effectiveness of our method in some realistic scenarios.**

Keywords - IoT; Reliability; Cumulative failure rate; Operating history; Multiple Players

I. INTRODUCTION

The concept of Internet of Thing (IoT) has been widely penetrating. According to [1], the number of devices which are available for mobile access is expected to grow to approximately 50 billion units (6.58 units per user) by 2020. As IoT-based systems are becoming more indispensable, they should be more reliable to achieve sufficient service availability. This cannot be achieved without high reliability and dependability of *IoT devices* themselves.

Cumulative failure rate is often utilized as a device reliability index [2]. It is a probability of occurrence of failure in a certain time period starting from the time when the device becomes in operation. The cumulative failure rate is usually derived using the failure rate for every unit of time, which is defined as a ratio of the number of failed devices to the number of devices being in operation in the unit of time. Here, the devices being in operation may vary at every moment not only due to device failure but also due to operational activities such as new device installation, removal and replacement. Therefore, we have to trace the operating history of each device to calculate the cumulative failure rate.

However, in order to obtain the operating history of each device, it is required to obtain the occurrence dates of device-associated events such as installation of the device, suspension and resumption of device utilization and failure. If the device manufacturers (simply called *vendors*) themselves provide services (this way of service provision is called *vertically integrated business model* [3]), such information can be obtained easily as it is managed in a single place. In contrast, in *horizontal specialization business model* [4] where service providers (simply called *providers*) purchase the devices from vendors and use them (this style is often seen in smart meter services and the Internet access services), device operating history is owned and managed partially and uncoordinatedly by multiple business operators called *players*, which will cause a significant issue in building a single, consistent view of operating history.

For example, in smart meter services, an electric company (i.e., a provider) purchases power meters in bulk from a vendor, lends them to subscribers, and stocks the rest as spare ones. When a power meter becomes out of order, the provider supports to replace it and orders a repair service to the vendor. Then, the vendor is able to manage the product-related information such as the production date and model number of the meter as well as the failure-related information such as the date and reason of failure and the repair process. However, the device operating status (e.g., operating start date) is not observed by vendor. Meanwhile, the provider has to manage the subscriber information including the asset information (e.g., the current meter location). Hence, it is not necessary to trace back the information about failure and others. Consequently, in order to obtain a consistent history of meters, it is necessary to design a method of aggregating the management information which are separately and uncoordinatedly managed by multiple players to enable calculation of the cumulative failure rates.

In this paper, we propose a method of calculating the cumulative failure rate, which is an important reliability index that represents device reliability. We assume that services are provided (i) using a large quantity of homogeneous devices and (ii) following the horizontal specialization business model where multiple players are involved and management information are owned separately and uncoordinatedly by them. Then, the method aggregates and analyzes those distributed information to derive the operating history of each IoT device to enable calculation of cumulative failure rates.

The contributions of this work are three-hold. Firstly, we deal with a significant issue of IoT device management through real case studies (based on our business experience) on how we grasp and measure the device reliability, which is mandatory for maintaining the quality of large-scale IoT service infrastructure operated by multiple players in the horizontal specialization business model. Secondly, we

 ISBN: 978-1-61208-450-3

propose a method to obtain the operating history of each IoT device from various types of management information. We note that calculating the cumulative failure rate using complete device history is usual in device management, but it is not straightforward taking those devices which are often replaces, repaired and reused at different times and locations into account. Finally, we present the experimental result of measuring the accuracy of cumulative failure rates with realistic scenarios where a part of information is missing, a situation that often occurs in the real environment.

This paper is organized as follows. Section II summarizes related work and Section III introduces a service scenario in IoT infrastructure with multiple players. Section IV presents our method and experimental results are shown in Section V. We conclude this work in Section VI.

II. RELATED WORK

There have been various activities on evaluating product quality and device reliability [5]-[8]. Several studies on Operation And Management (OAM) issues of IoT devices in IoT service infrastructure [9]-[11] have also been conducted.

References [5][6] present evaluation methods at the design or production phase of devices, where the cumulative failure rate is estimated by modeling the occurrence of major failures at the component level of devices. Reference [5] focuses on Hot Carrier Injection (HCI) and Time Dependence Dielectric Breakdown (TDDB) of N-channel Metal Oxide Semiconductor (NMOS) as a major failure factor. Reference [6] discusses how to determine a new parameter from failure factors observed in the field, e.g., electrostatic discharge inrush current, to combine with a conventional estimation method for more accurate cumulative failure rate at the product design phase. These approaches assume that all information elements, which are necessary for calculating the cumulative failure rate, are maintained in the vendor manufacturing the devices, and it is not considered that the number of devices varies by factors other than failures.

On the other hand, [7][8] propose evaluation methods for product reliability based on the observation of each device's operation history considering device changes due to non-failure, which is not taken into account in the existing work [2][12]. Specifically, the failure rate is calculated by the number of devices and time differentiation of the cumulative number of failure devices at the given timing of elapsed time.

All the above assume a vertical integration structure in which all the information elements for calculating the cumulative failure rate (or a similar index) are maintained by only one player. In contrast, we are focusing on IoT service infrastructure in which multiple players (e.g., providers and vendors) are collaboratively involved. In such a horizontal specialization structure, the followings should be done to manage the product reliability of IoT devices for realizing dependable infrastructure; 1) coordinating the management information provided by each player, 2) extracting and deriving information elements and 3) reconstructing the device operation history from the information elements. As far as we know, this is the first activity focusing on IoT device management with multi-player issues.

Meanwhile, there have been many approaches so far toward IoT device applications [9]-[11], which basically focus on the management and configuration of remote sensor devices over the Internet. Therefore, they do not deal with the IoT device management issues.

III. SERVICE SCENARIO

In this paper, we assume IoT infrastructure with multiple players in the horizontal specialization business model. Under this assumption, we explain the device operating and management information that are separately and uncoordinatedly managed by multiple players. The scenario is based on our own experience, so everything here is likely to occur in the real world business.

As explained briefly in Section I, we target such a service provider as an electric company or a network provider that purchases the devices in bulk from an IoT device vendor and lends them to subscribers (users). Figure 1 illustrates the interactions between each player and users. We explain the service provision scenario using this figure.

(1) IoT Device Purchase and Stocking: The provider purchases IoT devices from the vendor and stocks them as spares. Lending an IoT device from the provider to a user and returning it by the user due to cancellation is conducted via the provider's warehouse. The provider records the *current* location of the purchased IoT devices in asset management information. The vendor records the product-related information such as the shipping date and model number of IoT devices in shipment management information.

(2) Service Startup: The provider creates the contract-related information for every user and manages it. The provider lends an IoT device to a user, starts the service and records the service start date in user contract information.

(3) IoT Device Failure and Replacement: When an IoT device fails at a user location, the provider sends an alternative IoT device to the user. The user sends the failed IoT device to the address of the repair service. Since the vendor repairs the failed IoT device, the user sends it back to the vendor directly to optimize the transport route. The vendor repairs the failed IoT device and records the failure-related information such as date and model number (or send-back date or receiving date as the date of failure). After the repaired IoT device is sent from the vendor to the provider, it is stocked in the warehouse. The provider updates the records related to these two devices (i.e., the current locations of failed and alternative devices).

(4) Service Cancellation: When a user cancels its contract, she/he returns her/his IoT device to the provider. The provider re-stocks it in the warehouse and updates the current location information of the IoT device. Furthermore, the service end date of this user is recorded in the contract-related information.

In summary, the provider maintains a) contract information with users and b) asset management information of IoT devices, the vendor maintains c) shipment-related information of IoT devices and d) the failure-related information. Here, b) is usually sufficient for asset management by the provider. This is because the provider

 ISBN: 978-1-61208-450-3

does not care about whether an IoT device was installed at different locations in the past.

On the other hand, as described in Section I, it is required to obtain the occurrence dates of device-associated events such as installation of the device, suspension and resumption of device utilization and failure to calculate the cumulative information such as the service start date and suspension and resumption of the device date in this scenario, and the provider does not observe the information such as the failure date. Therefore, each player cannot collect and build complete device-associated information. This is our motivation to provide a method to build complete operating history of each IoT device from such partial, distributed operating and management information as indicated by the above a) to d).

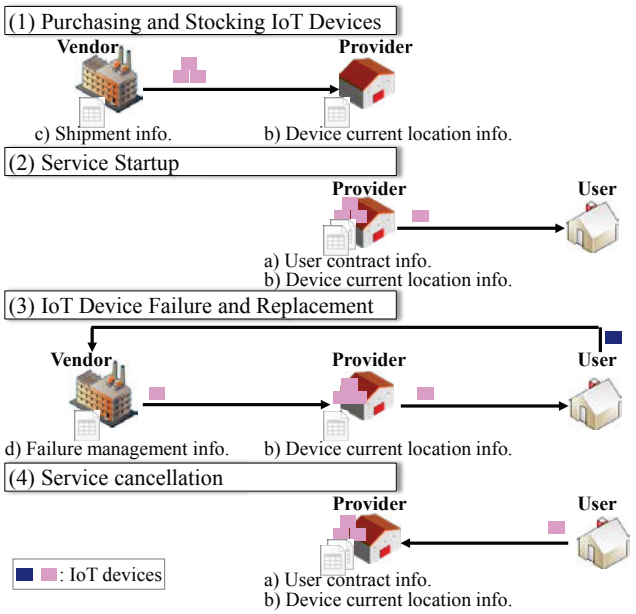


Figure 1. Interaction among multiple players and user.

Firstly, from a) and b), we try to obtain List-A of Table I with its created date *Tc*. Each pair of the sequence number *SN* and its location *L* can be obtained from b). This *L* is matched with that in a) to associate this pair with the service start date *T1* contained in a). If this device has not been failed since the day of initial installation at a user, we can obtain the history indicating that device *SN* has been working without failure between *T1* and *Tc*, which results in the fact that the operation start date *T4* is *T1* (*T4*=*T1*). Meanwhile, if there was a failure, *T1* is set to the date when an alternative device is started working at location *L*. In this case, *T4*, the operating start date of the original device *SN,* is left *unknown*.

Secondly, we try to obtain List-B of Table I from a) and d). In this scenario, the vendor records the location where the failure occurred (this kind of information is generally useful for such vendors which need some statistics of failure occurrence patterns). Here, we should consider how we will obtain column *T5*, which is the operation start date of each failed device. To do this, we associate date *T2* of failure, the sequence number *SN* and location *L* with contract information of a). Then, we obtain *T5* and the history indicating that device *SN* installed at location *L* had been working from *T5* until *T2* and then failed at *T2*.

Moreover, a) contains the service end date *T3* and the service start date *T1*. If we have sequence number *SN* of the device that was returned from location *L,* we can obtain List-C of Table I containing *T6,* the operation start date of the returned device. We note that the provider may not be motivated to record sequence number *SN*. Similarly with the List-A case, from this List-C, we can obtain the history indicating that the returned device *SN* had been working from *T6* until *T3* without failure. Under a certain condition, *T6* is equal to *T1*.

TABLE I. SERVICE OPERATING AND MANAGEMENT DATA

(LIST-A) CURRENT DEVICE LIST (MANAGED BY PROVIDER)

List created date (=Today) (*Tc*) : 2014/10/01			
Location (*L*)	Service start date (*T1*)	Current Device (*SN*)	Operating start date of current device at *L* (*T4*)
1	2014/01/01	a	-
3	2014/05/01	b	-
4	2014/03/01	c	-
:	:	:	:

(LIST-B) FAILED DEVICE LIST (MANAGED BY VENDOR)

Location (*L*)	Failed Date (*T2*)	Failed device (*SN*)	Operating start date of failed device at *L* (*T5*)
1	2014/02/01	a	-
3	2014/04/01	a	-
1	2014/04/01	b	-
:	:	:	:

(LIST-C) RETURN DEVICE LIST (OUT OF MANAGEMENT BY PROVIDER)

Location (*L*)	Return date (*T3*)	Return device (*SN*)	Operating start date of return device at *L* (*T6*)
3	2014/03/01	c	-
5	2014/06/01	d	-
2	2014/07/01	e	-
:	:	:	:

(: unknown)

In the next sections, we present how *T4*, *T5* and *T6* are obtained using List-A, B and C, and how the cumulative failure rate is calculated using the history.

IV. PROPOSED METHOD

A. Overview

In this section, we explain how to obtain the cumulative failure rate of IoT devices whose management information is maintained separately and uncoordinatedly by multiple players. Our proposed method consists of the following three Steps;

Step1: Reconstructing the operating history of each IoT device,
Step2: Counting the operating days, and
Step3: Calculating the cumulative failure rate.

Specifically, our proposed method basically uses List-A and B for reconstructing the operating history, and List-C as well as (if exists). Note that even without List-C, the method can reconstruct the history but some error may occur because *T3*, and *T6* in List-C are not plotted on the time-sequence diagram (See Figure 2 in Section IV-B). We numerically evaluate the impact of such error in Section V.

 ISBN: 978-1-61208-450-3

B. Design Details

Step1: Reconstruct the operating history of IoT device.

Step1-1) Create time-sequence diagram per location.

(1) First, the time-sequence diagram per location is created as shown in Figure 2 (i) where x- and y-axes are # of days passed (denoted as *T*) from the reference date "0" and location *L* (1, 2, …), respectively. Current date *Tc*, failed date *T2* and return date *T3* in List-A, B, and C are plotted as square boxes on the diagram at (*x*,*y*)=(*Tc*/*T2*/*T3*, relevant *L*), respectively. Note that for easy understanding, in Figure 2, we assign a numeral number *j* to each plot as ID. It is denoted inside the square box corresponding to the plot.

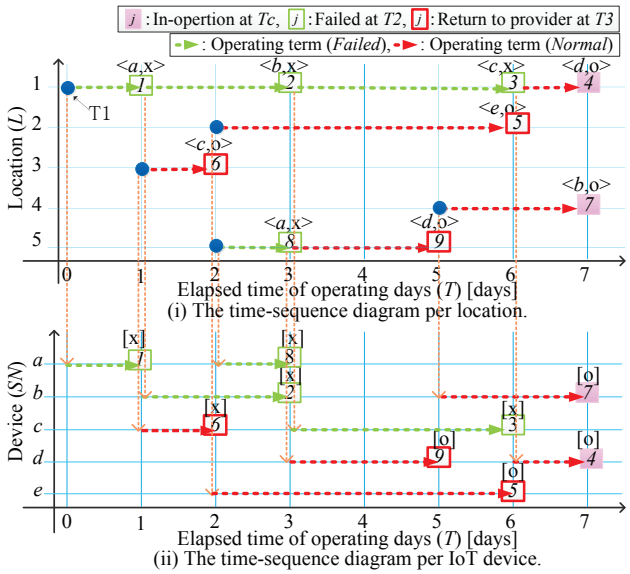


Figure 2. Time-sequence diagrams for reconstructing the operating history of each IoT device.

(2) For each plot *j*, device *SN* and device operating status x="*Failed*" or o="*Normal*" at the relevant date are associated as its attribute. For instance in Figure 2 (i), plot *j*=8 with <*a*,x> (at *T*=3 and *L*=5) means that the device *a* was "*Failed*" at location *L*=5 (then it was sent back to the vendor for repair). Plot *j*=7 with <*b*,o> means that the device *b* was "*Normal*" at *L*=4 (therefore it is in-operation now (*T*=7)). Plot *j*=9 with <*d*,o> means that the device *d* was "*Normal*" at *L*=5 (because it was returned to the provider without failure due to user cancellation at *T*=5).

(3) Service start date *T1* at location *L* in List-A is plotted on the diagram.

(4) We assume that the failed device is replaced to another device on the same day for non-stop service. Along the time-sequence of each location *L*, the plots on it are traced back from the current date *Tc* (*T*=7) to the reference date (*T*=0) in order to determine the start date (referred to as S_j) of each plot *j* at the location. The date of *j*'s previous plot is regarded as the start date of *j*. For example, the start date of device *b* at plot *j*=2 ((*x*,*y*)=(3,1)) is determined as S_2=1 because its previous plot (*j*=1) is at *T*=1 ((*x*,*y*)=(1,1)).

(5) The operating term for each plot *j* can be extracted as the term from S_j to *T* of plot *j*. For example, device *b* at plot *j*=2 is operated from S_2=1 to *T*=3 so the term is 2 days. As another example, device *b* at *j*=7 ((*x*,*y*)=(7,4)) is operated from S_7=5 to *T*=7 (now in-operation) so the term is also 2 days.

Step1-2) Transform time-sequence diagram per location to per IoT device.

(1) The time-sequence diagram per IoT device (see Figure 2 (ii)) is transformed from Figure 2 (i). At first, each plot *j* in Figure 2 (i) is re-plotted on Figure 2 (ii) according to the device *SN* in its attribute. Note that the device operating status x/o in its attribute is inherited. For instance, plot *j*=2 with <*b*,x> ((*x*,*y*)=(3,1)) in Figure 2 (i) is re-plotted to *j*=2 with [x] ((*x*,*y*)=(3,*b*)) in Figure 2 (ii).

(2) For each plot *j*, the relevant operating term from S_j to *T* of the plot *j* is drawn on Figure 2 (ii). It is easy to obtain each IoT device's operating history by collecting operating terms per device from Figure 2 (ii). For instance, the operating history of device *b* includes two operating terms, i.e., S_2=1 to *T*=3 (*Failed*) and S_7=5 to *T*=7 (*Normal*).

Step2: Counting the operating days.

Two types of operating days are counted per IoT device from the operation histories. The first type is referred as "*Failed days*" (*P*) which is ended with a plot derived from *T2*, i.e., failed date. The second type is referred to as "*Normal days*" (*Q*) which is ended with a plot derived from *T3* or *Tc*, i.e., return or current date without failure. For counting the operating days of each IoT device, an operation term of the device is selected in chronological order and checked whether the term is ended by *T3* or not. If so, the term should be concatenated to the next operating term (if any) as a single piece of operating days. For example in Figure 2 (ii), the device *b* is set *P*=2 and *Q*=2, while the device *c* is set *P*=4(=1+3), and the device *d* is set *Q*=3(=2+1). *P* and *Q* of each device are shown in Table II.

TABLE II. THE OPERATING DAYS FOR EACH *SN* IN FIGURE 2 (ii)

SN	# of operating days (# of terms)	Operating days [days]	
		1st	2nd
a	2 (2)	*P*=1	*P=1*
b	2 (2)	*P*=2	*Q=2*
c	1 (2)	*P*=4(=1+3)	-
d	1 (2)	*Q*=3(=2+1)	-
e	1 (1)	*Q*=4	-

Step3: Calculating the cumulative failure rate.

From both *Failure days* and *Normal days* in Step2, the cumulative failure rate is calculated. Let *f*(*x*) denote the failure density function, the failure occurrence probability until time *i* has passed, i.e., the cumulative failure ratio *F*(*i*), is expressed in Eq. (1) [2].

$$F(i) = \int_0^i f(x)dx \quad (1)$$

We can approximately obtain the following difference equation by differentiating Eq. (1) and substituting infinitesimal *di* to the unit time (a day).

$$F(i) - F(i-1) = f(i) \quad (2)$$

Here, let $\lambda(i)$ denote the failure rate per unit time. Since $f(i) = \big(1 - F(i-1)\big) \cdot \lambda(i)$, E.q. (2) is expressed as;

$$F(i) = F(i-1) + \big(1 - F(i-1)\big) \cdot \lambda(i) \qquad (3)$$

where

$$F(0) = 0, \qquad \lambda(i) = \frac{n(i)}{N(i)+n(i)} \quad (i = 1, 2, \dots) \qquad (4)$$

and $n(i)$ is the number of failed devices (P = i-1) at day i, $N(i)$ is the number of in-operation devices at the end of day i. From the above discussions, the cumulative failure rate can be calculated from the operating history.

In addition, the number of returned devices, which is suspended at day i, is given by $N(i\text{-}1) - (N(i) + n(i))$. Assuming that $\lambda(i)$ is the same regardless of the devices, the cumulative failure rate is not affected even if suspended devices exist.

V. EXPERIMENTAL EVALUATION AND DISCUSSION

We verify that the proposed method expressed in Section IV can reconstruct operating histories and calculate the cumulative failure rate. In addition, we evaluate the accuracy of the cumulative failure rate when some information elements are missing.

A. Experimental Setup

In order to validate the effectiveness of our proposal, we develop the simulator implementing the proposed method. The input data set for this simulator consists of List-A, B, and C (if any) without *T4, T5*, and *T6*. From these input data set, the simulator complements the unknown fields (*T4, T5*, and *T6*), then reconstructs operating histories and calculates the cumulative failure rate. This simulator is a Ruby program with approximately 17,000 lines, executing on a PC with the following specifications; CPU: E5-2650L v2@1.70GHz, memory: 126GBytes, OS: CentOS 6.6, Ruby 1.9.3.

Evaluation data sets as shown in Table III are arranged with various return rates R and failure rates U, both of which follow uniform distribution irrespective of T. Simulation days T is 1,826 days (= 5 years), the maximum number of devices is 70,000 [units], the maximum number of locations is 10,000. We assume no IoT devices are in-operation at T = 0, and the replacement of failed device is finished on the same day as the failure occurs. For accurate evaluation results, we arranged 180 data sets in total, because 18 R/U pairs are specified and 10 random data sets are generated per R/U pair. Note that these data sets are given as List-A, B and C. At first, a data set consists of all the information elements in List-A, B and C is generated (we call it "reference data set"). Then, an evaluation data set is created from it by omitting unknown fields, i.e., *T4*, *T5* and *T6* or *T4*, *T5* and whole List-C, according to the case in Table IV.

TABLE III. PARAMETERS OF CREATING EVALUATION DATA

Parameters	Values
Failed rate U [%/day]	0.2, 0.5, 0.8
Return rate R [%/day]	0, 0.2, 0.4, 0.6, 0.8, 1.0
The number of simulation days T [days]	1,826(= 5 years)
The number of devices [units]	15,000～70,000
The number of locations [locations]	100～10,000

TABLE IV. VERIFICATION CASES

Case #	List-C management / non-management	unknown data
Case1	management (=use List-C)	*T4*, *T5*, *T6*
Case2	Non-management (=not use List-C)	*T4*, *T5*, List-C*

**T6*: unknown because of List-C unmanaged

B. Verification of proposed method

We verify that the proposed method can complement the unknown fields in List-A, B, and C in Case1 and Case2 by comparing with the reference data sets. We also reconstruct operating histories and calculate cumulative failure rates from all the evaluation data sets.

As a result, we confirm that the simulator successfully completes the above processes for any data sets in any cases. In Case1, all *T4*, *T5* and *T6* of event start date are completely matched with those in the reference data sets. In contrast, in Case2, *T4* and *T5,* which are operating start dates of failed and current devices respectively, are different from those in the reference data sets. This is because a part of these start dates are changed due to the lack of List-C. As one of the examples at R=1.0, 18.1%, 23.6% and 24.9% of these start dates, i.e., *T4* and *T5*, are changed in average when U=0.2, 0.5 and 0.8, respectively.

On the other hand, the example computation time to obtain the operating history and the cumulative failure rate is approximately 3 [min] and 1 [min], respectively, in the case of U=0.8 and R=1.0 with 70,000 IoT devices.

C. The Effect of cumulative failure rate by return rate R

Figure 3 shows the cumulative failure rate $F(i)$ in each failure rate U where the return rate R varies from 0.0% to 1.0% at 0.2% intervals. Each plot indicates the average of $F(i)$ values individually calculated from 10 random data sets arranged per R/U pair.

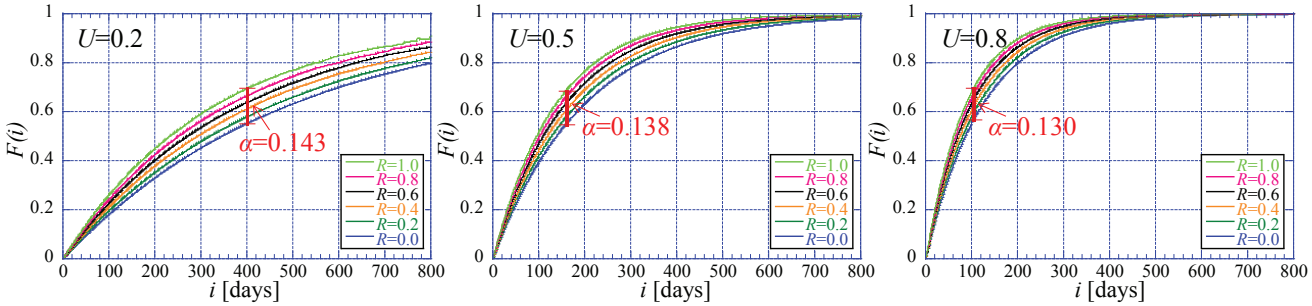


Figure 3. Cumulative failure rate $F(i)$ vs. operating days i varied return rate R of each.

It is clear in Figure 3 that $F(i)$ increases in proportion to R irrespective of U and that the larger R becomes, the more rapidly the cumulative failure rate $F(i)$ increases. As for the errors of $F(i)$ (referred to as α) between R=0.0 and 1.0, α =0.143 at i=400, α=0.138 at i=159, and α=0.130 at i=105, respectively. So the error α decreases with increase of U.

Figure 3 also indicates that our method conservatively underestimates the cumulative failure rate. Hence from the provider's perspective, the calculated rate can be useful when the provider discloses it to the vendor for encouraging more improvement on the product quality and reliability of IoT devices. However, in the reverse direction from the vendor to the provider, such under estimation may mislead what each player should do next. So, the calculated rate should be interpreted carefully according to the player's role.

D. On addition of information elements after service start

In this scenario, we assume that each player is dedicated to playing his role and does not have any incentive to maintain extra management information beyond his role. However, in the real world, it is probable that either player may add some management information due to emerging new operational requirements after the service start, e.g., similar service infrastructures and their providers are merged into one.

Here, we briefly and qualitatively discuss on how to handle such management information change, especially in the case that some useful information elements can be obtained after a certain date. For example in Sec. III, it is considerable that the service was started with a very small number of users, it was not so important for the provider to improve product reliability of IoT devices at first. However, as increasing of IoT devices after time goes by, the provider want to improve the product reliability of IoT devices more so that he start maintaining List-C from a certain date ($T=Y$)

In such a case, return date *T3* in List-C is started from Y and there are no previous records before it, i.e., from $T=0$ to $Y-1$. For calculating the cumulative failure rate, we can choose one of the following three options.

1) The calculation is conducted using recorded *T3* ($T3 \geq Y$) only, assuming that no return event, i.e., service cancelation, occurs at $T<Y$.
2) The calculation is conducted after complementing *T3* at $T<Y$ based on R calculated from recorded *T3* ($T3 \geq Y$).
3) The calculation is conducted without List-C.

Among above three methods, Option 3 is equivalent to the result of $R=0.0$ in each U in Figure 3. According to the results in Figure 3, the cumulative failure rate is qualitatively expected Option 3 > Option 1 > Option 2. The quantitative evaluation of Options 1 and 2 is left as our future work.

VI. CONCLUSION

In this paper, we have proposed a method of calculating the cumulative failure rate in IoT service infrastructure operated by multiple players such as service providers and device vendors in the horizontal specialization business model. The method aggregates and analyzes distributed information to derive the operating history of each IoT device to enable calculation of cumulative failure rates. We have verified that the proposed method can derive operating histories and calculate the cumulative failure rate. In addition, we have evaluated the accuracy of the derived cumulative failure rates when some information about device operation are missing. Furthermore, we have discussed cases where the missing information are added after the service is started to see the performance of our method in real world cases.

We are now planning to apply our method to more different cases. Based on our business experience, there are a variety of scenarios where management information are distributed and unmanaged. We believe that we have shown the applicability of our method by introducing well-seen, representative cases in this paper, but examination of our approach in a variety of scenarios is part of our future work.

REFERENCES

[1] Cisco Systems Inc., "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, accessed Jan. 8, 2016.

[2] W. Zheng, W. Zengquan, and W. A-na, "Failure Rate Calculating Method of Components Based on the Load-strength Interference Model," Proc IEEE Industrial Engineering and Engineering Management IEEM 2010, pp.783-787, Dec. 2010.

[3] OPS rules, "Vertical vs. Horizontal Integration: Which is a better Operations Strategy?," Sep. 2012, http://www.opsrules.com/supply-chain-optimization-blog/bid/241648/Vertical-vs-Horizontal-Integration-Which-is-a-Better-Operations-Strategy, accessed Jan. 8, 2016.

[4] Z. Yu, "IT, Production Specialization, and Divison of Labor: A Smith-Ricardo Model of International Trade," Carleton Economic Paper, Jun. 2003, http://carleton.ca/economics/wp-content/uploads/cep03-06.pdf, accessed Jan. 8, 2016.

[5] Z. Zhou, X. Liu, Q. Shi, Y. En, and X. Wang, "Failure Rate Calculation for NMOS Devices under Multiple Failure Mechanisms," Proc International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA), pp.362-365, Jul. 2013.

[6] T. Tekcan, G. Kahramanoglu, and M. Giinduzalp, "Determining Reliability by Failure Rate Estimation via a New Parameter," Proc Reliability and Maintainability Symposium (RAMS), pp.1-7, Jan. 2012.

[7] H. Funakoshi and T. Matsukawa, "A Failure Rate Estimation Considering the Change in the Number of Equipments," IEICE NS2009-17, pp.1-6, 2009 (in JAPANESE).

[8] H. Funakoshi and T. Matsukawa, "A failure Rate Estimation Considering the Change in the Number of Equipments," IEICE Trans. B Vol. J93-B No.4, pp.681-692, 2010 (in JAPANESE).

[9] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V.C.M. Leung, "Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things," in IEEE Internet of Things Journal, vol.2, no.5, pp.402-411, Oct. 2015.

[10] C. Zhou and X. Zhang, "Toward the Internet of Things application and management: A practical approach," in IEEE WoWMoM 2014, pp.1-6, 2014.

[11] S.N. Han, S. Park, G.M. Lee, and N. Crespi, "Extending the Devices Profile for Web Services Standard Using a REST Proxy," in IEEE Internet Computing, vol.19, no.1, pp.10-17, Jan.-Feb. 2015.

[12] M. Xie, Y. Tang, and T. N. Goh, "A modified Weibull extension with bathtub-shaped failure rate function," Reliability Engineering and System Safety, 2002, 76(3): 279-285.

 ISBN: 978-1-61208-450-3

Performance Evaluation of the Opportunistic Spectrum Access in a Cognitive Radio Network with Imperfect Sensing

Branislav Edgar Feijó Couceiro and José Marcos Câmara Brito
Instituto Nacional de Telecomunicações-INATEL
Santa Rita do Sapucaí, Brazil
branislavcouceiro@gmail.com, brito@inatel.br

***Abstract*—Cognitive radio networks have emerged as a promising technology to shift the actual paradigm of spectrum scarcity by allowing the cognitive users (secondary users) to use the licensed spectrum on an opportunistic basis. A continuous-time Markov chain (CTMC) can be used to model the *opportunistic spectrum access* (OSA) in a cognitive radio network with imperfect sensing (false alarms and misdetections). In this paper, an analytic model is developed and the performance of the system is evaluated and the numerical results are presented in terms of capacity (average number of calls completion per time unit), blocking probability, force termination probability and spectrum utilization.**

Keywords—performance evaluation, cognitive radio networks, continuous-time Markov chain (CTMC) model, imperfect sensing.

I. INTRODUCTION

Cognitive radio technology offers an alternative spectrum access, which could result in an effective use of the spectrum. In cognitive radio networks, there are two types of users: primary users, who own (have license for the use of) a frequency band, and secondary users, who attempt to use the licensed frequency band in an opportunistic way. Primary users have priority over secondary users and secondary users (unlicensed) can access the licensed spectrum band if the owner of that band is absent. In order to access those frequency bands without interfering with primary users transmissions, secondary users must accurately detect the presence of primary users by sensing the spectrum. Different sensing techniques can be used to perform spectrum sensing [10], but sensing techniques cannot provide perfect sensing. So, false alarms and misdetection cannot be fully avoided. False alarms happen when a channel is free and the cognitive user classifies that channel as busy, and misdetections happen when a busy channel is classified as free. In [2], analysis of the access of cognitive radio networks was studied using continuous time Markov chain (CTMC), but the effect of imperfect sensing was not taken into account. A similar state diagram based approach was taken in [6]-[8] for modeling dynamic spectrum access with channel assembling, but, also perfect sensing accuracy was assumed. In [3], the effect of imperfect sensing is studied alongside with the opportunistic spectrum access in a cognitive radio network. Explicit expressions for state dependent transition rates were presented for the specific case of three channels. However, a generalization of the study for more channels was not done.

The prominent feature of the state diagram in [3] is that it has state-dependent transition rates at the nodes. These transitions are found by going through all possible transitions from a state taking into account the channel state and the sensing decision.

In this paper, we generalize the approach used in [3] and provide expressions to compute the state transition rates in a generic way, which can be used for any numbers of channels. Also, additional performance metrics are presented to analyze the performance of the system. The rest of this paper is organized as follows. Section II describes the system model and assumptions. Section III describes the CTMC analysis. Section IV goes into details with respect to the network performance analysis. Numerical results are presented in Section V. Finally conclusions are presented in Section VI.

II. SYSTEM MODEL AND ASSUMPTIONS

Similar to [3], we consider that there are N channels available, and these channels can be accessed by primary and secondary users, with primary users having priority over the secondary users. Primary and secondary calls arrive with rates λ_1 and λ_2 respectively, and are completed with rates μ_1 and μ_2, respectively. In order to find the same theoretical solutions as in [3], we use the exact same assumptions. These assumptions are:

- When the primary user arrives to a channel occupied by a secondary user, the secondary user will always notice the primary user and will leave the channel. After this, the secondary user starts to search for a new channel. During this phase, the secondary user will perform detection on the remaining channels in random order until it finds a free channel or all channels are determined to be busy. A free channel is decided to be occupied with false alarm probability P_F and an occupied channel is determined to be free with misdetection probability $P_M = 1 - P_D$, where P_D is the detection probability, which refers to a probability that an occupied channel is correctly detected as busy.

 ISBN: 978-1-61208-450-3

- The sensing time and the time to perform the spectrum handover is considered negligible. So, all states transitions are instantaneous.
- A secondary user always knows the channels occupied by other secondary users and will not sense or use them. Therefore, there is no collision between secondary users.
- A primary user knows the channels occupied by other primary users so that there will be no collision between primary users.
- In case of collision between primary and secondary users, both colliding users withdraw from the channel. This means that both transmissions will be lost. Note that the collision when primary user comes to a channel used by a secondary user is assumed to be short and does not cause the primary user to leave the channel.
- The search order for new free channel is random and equally probably. The search stops after an idle channel is found or all channels are found to be occupied.

III. CTMC ANALYSIS

A two-dimensional Markov chain is used to model the system. Let $x=\{i,j\}$ be the general state representation, where i represents the number of primary users' calls in the system and j the number of secondary users' calls in the system. For example, $x=\{2,1\}$, refers to the state with two primary users and one secondary user. Let N be the number of channels available in the system. Therefore, the set of feasible states of the system is denoted as $S=\{x|\ 0 \le i \le N,\ 0 \le j \le N,\ 0 \le i + j \le N\}$.

A. State Transition Rate

The state-dependent transition rates are derived considering all possible events that can trigger state transition and all possible sensing decisions.

For example, the transition from the state $x=\{1,0\}$ to the state $x=\{0,0\}$, considering 3 channels [3], occurs in four different scenarios:

1) First, it can happen if the primary user regularly completes its call with rate μ_1;

2) It can happen if a secondary user arrives to the channel occupied by the primary user, fails to detect the presence of the primary user (with probability P_M) and transmits on that channel resulting in a collision. In this case, both transmissions are lost taking the chain to the state $x=\{0,0\}$. Since the secondary user searches for a new channel randomly and with equal probability, the transition rate in this scenario is $1/3\lambda_2 PM$;

3) Another possibility is if the secondary user arrives in one of the two idle channels (with 2/3 probability), but, after sensing, it decide that the channel is busy (with probability P_F). In this case, if the secondary user goes to the channel occupied by the primary user (with probability 0.5) and fails to detect the presence of the primary user (with probability P_M) and transmits on that channel, the secondary user will collide with the primary user and take the Markov chain to the state $x=\{0,0\}$, with rate $2/3\lambda_2 PF1/2PD$.

4) The last possibility is if the secondary user arrives in one of the two idle channels (with 2/3 probability), but, after sensing, it decides that the channel is busy (with probability P_F), then, the secondary user searches another channel and goes to the other idle channel (with probability 0.5) and decides that this channel is also busy (with probability P_F), in this case, the secondary continues his search and goes to the channel that is occupied by the primary user, fails to detect the presence of the primary user (with probability P_M) and transmits on that channel resulting in collision and both users withdraw from the channel and the Markov chain will go to state $x=\{0,0\}$ with rate $2/3\lambda_2 P_F 1/_2 P_F P_M$. So, the transition from state $x=\{1,0\}$ to state $x=\{0,0\}$ will be:

$$i\mu_1 + \lambda_2 \frac{1}{3} P_M + \lambda_2 \frac{2}{3} P_F \frac{1}{2} P_M + \lambda_2 \frac{2}{3} P_F \frac{1}{2} P_F P_M \qquad (1)$$

Figure 1 shows the Markov chain with its explicit transition rate for the case of three channels reproduced from [3].

The same approach should be taken to determine the transition rate from any pair of states directly connected. As we can see, it can be an exhaustive work, especially if the number of channels is large. In this paper, we developed generic expressions which can be used to determine the transition rate from one state to another for any number of channels N. These expressions are presented in Table I.

B. Transition Matrix and Stationary Probability

Let Q be the transition matrix of the CTMC. We determine the total transition rate from state a to state b, which is the summation of transition rate from state a to state b considering all possible user activities for all a,b ($a \neq b$) ϵ S namely q_{ab}. The diagonal elements in Q, i.e., q_{aa}, $a \epsilon$ S are found as [8]:

$$q_{aa} = -\sum_{b \in S, b \neq a} q_{ab} \qquad (2)$$

The stationary probabilities, $\pi(x)$ can be computed from the global balance equations and the normalization equation, given as:

$$\pi Q = 0, \quad \sum_{x \in S} \pi(x) = 1 \qquad (3)$$

where π is the steady state probability vector.

Once the steady state probabilities are determined, the performance of the system can be evaluated by different parameters, as presented in the next section.

IV. PERFORMANCE ANALYSIS

The analysis presented in [3] considers only two parameters, the primary user termination probability and the probability

TABLE I. TRANSITIONS RATE FROM A GENERIC STATE (i,j)

Dest. state	Conditions	Transition Rate
$(i+1, j)$	$i < N$	$\frac{\alpha}{N-i}\lambda_1 + \frac{j}{N-i}\lambda_1\left[\sum_{t=1}^{\alpha}\binom{\alpha}{t}(1-P_F)^t P_F^{\alpha-t}\left(\sum_{r=0}^{i}\binom{i}{r}P_D{}^r P_M{}^{i-r}\frac{t}{t+(i-r)}\right)\right]$
$(i-1, j)$	$i > 0$	$i\mu_1 + \lambda_2\left[\sum_{t=1}^{i}\binom{i}{t}P_M{}^t P_D{}^{(i-t)}\left(\sum_{r=0}^{\alpha}\binom{\alpha}{r}P_F{}^r(1-P_F)^{\alpha-r}\frac{t}{t+(\alpha-r)}\right)\right]$
$(i, j+1)$	$(i+j) < N$	$\lambda_2\left[\sum_{t=1}^{\alpha}\binom{\alpha}{t}(1-P_F)^t P_F{}^{[\alpha-t]}\left(\sum_{r=0}^{i}\binom{i}{r}P_D{}^r P_M{}^{i-r}\frac{t}{t+(i-r)}\right)\right]$
$(i, j-1)$	$j > 0, i = 0$	$j\mu_2$
$(i, j-1)$	$j > 0, i > 0$	$j\mu_2 + \frac{j}{N-i}\lambda_1\left[\sum_{t=1}^{i}\binom{i}{t}P_M{}^t P_D{}^{i-t}\left(\sum_{r=0}^{\alpha}\binom{\alpha}{r}P_F{}^r(1-P_F)^{\alpha-r}\frac{t}{t+\alpha-r}\right)\right]$
$(i+1, j-1)$	$i < N, j > 0$	$\frac{j}{N-i}\lambda_1 P_F{}^{\alpha} P_D{}^i$

$\alpha = [N-(i+j)]$

that a secondary call is normally terminated, and only for a particular case (N=3).

In this paper, we introduced four new parameters to analyze the performance of the system:

a) Secondary blocking probability

b) Primary network capacity

c) Secondary network capacity

d) Spectrum utilization

Also, using the generic transition rates presented in Table I, we can compute the performance of the system (for our new four parameters and the two parameters presented in [3]) for any number of channels.

In this section, we presented the expressions to compute each performance parameter. Numerical results and discussions are presented in Section V.

A. Primary Network Performance Analysis

1) Primary user termination probability: As defined in [3], the primary user termination probability is the probability that a primary user call, which has not been blocked at start, is terminated due to collision with secondary users because of misdetection. This probability is computed by [3]:

$$P_{PT} = \frac{\left[\sum_{i=1}^{N}\sum_{j=0}^{N-1}\pi(i,j)\left(T_{(i-1,j)}^{(i,j)} - i\mu_1\right) + \sum_{i=1}^{N-1}\sum_{j=1}^{N-1}\pi(i,j)\left(T_{(i,j-1)}^{(i,j)} - i\mu_2\right)\right]}{\lambda_1\left(1-P_{b1}\right)} \quad (4)$$

where $\pi(i,j)$ is the state probability of the state (i,j) and $T_{(i-1,j)}^{(i,j)}$ is the transition from state (i,j) to state $(i-1,j)$, and P_{b1} is the primary user blocking probability given by:

$$P_{b1} = \pi(N, 0) \quad (5)$$

2) *Primary network capacity*: We define the term capacity as the average number of calls completed per unit time. Either (5) or (6) can compute the primary network capacity:

$$\rho_1 = \lambda_1(1-P_{b1})(1-P_{PT}) \quad (6)$$

$$\rho_1 = \sum_{x \in S} i\mu_1\pi(x) \quad (7)$$

where *i* is the number of primary user's calls in state *x*.

B. Secondary Network Performance analysis

1) Secondary User Blocking probability: A secondary user's call is blocked if a secondary user arrival does not take the system to a state (i,j+1). Therefore, the blocking probability can be expressed as:

$$P_{b2} = \sum_{x \in S}\pi(x)\left(1 - \frac{T_{(1,j+1)}^{(i,j)}}{\lambda_2}\right) \quad (8)$$

 ISBN: 978-1-61208-450-3

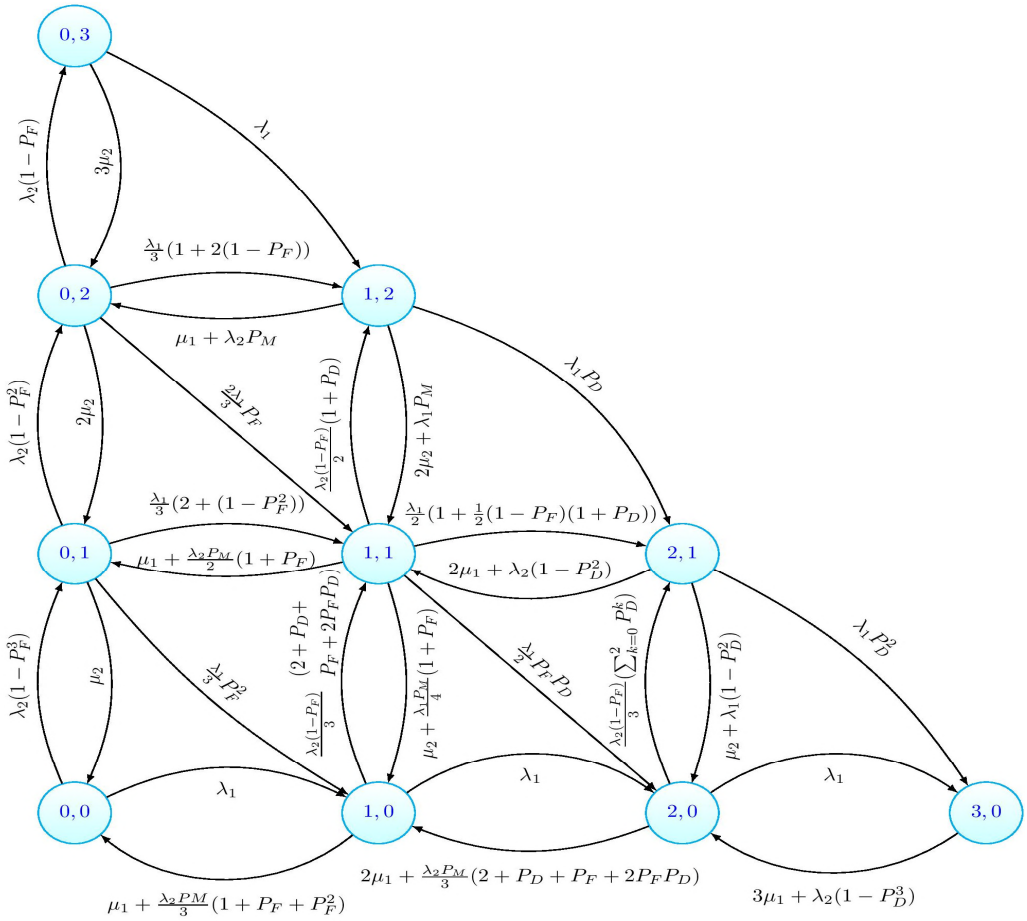


Figure 1 State diagram

2) Secondary user termination probability: We define the secondary user termination probability as the probability that a secondary user call, which has not been blocked at start, is forced to terminate before its transmission is finished. So, the secondary user termination probability can be expressed as:

$$P_{ST} = \frac{R_{int}}{\lambda_2(1-P_{b2})} \tag{9}$$

where Rint is the rate that a primary user preempts a secondary user's call in a way that the secondary user is forced to terminate.

Than, we have:

$$R_{int} = \left[\sum_{x\in S}\pi(x)(T^{(i,j)}_{(i,j-1)} - j\mu_2) + \sum_{x\in S}\pi(x)T^{(i,j)}_{(i+1,j-1)}\right] \tag{10}$$

Therefore, the probability that a secondary call is normally terminated can be expressed as:

$$P_{SNT} = (1-P_{b2})(1-P_{ST}) \tag{11}$$

3) Secondary network Capacity: We define this parameter as the average number of secondary user´s call completion per time unit. Either (10) or (11) can compute the secondary network capacity. Therefore, the secondary network capacity will be:

$$\rho_2 = \lambda_2(1-P_{b2})(1-P_{ST}) \tag{12}$$

$$\rho_2 = \sum_{x\in S} j\mu_2\pi(x) \tag{13}$$

where *j* is the number of primary user´s calls in state *x*.

C. Spectrum Utilization

We define the spectrum utilization as the average number of utilized channels over the total number of channels. The spectrum utilization of the cognitive radio network can be expressed as:

$$U = \sum_{x\in S}\pi(x)\frac{(i+j)}{N} \tag{14}$$

 ISBN: 978-1-61208-450-3

V. NUMERICAL RESULTS

In this section, numerical results are presented to illustrate the performance of the opportunistic spectrum access with respect to parameters presented in the previous section, taking into account the effect of the imperfect sensing.

In order to compare the results presented here with the results presented in [3], we opted to use the same values used in [3], for the following parameters: $\lambda_1 = 7$, $\lambda_2 = 3.5$, $\mu_1 = \mu_2 = 4$, $P_F = 0.15$ and $P_D = 0.713$.

In our analysis, we can define the number of channels as any value. To illustrate the results, we opted to use N=5 and N=10. We also used N=3 for the sake of comparison with the results in [3]. N=3 is the only value presented in [3].

1) Primary user termination probability: Figure 2 shows the termination probability of a primary user as the P_D varies. It can be seen that when the $P_D = 1$, the termination probability is zero ($P_D = 1$ means perfect detection and so there is no interruption from the secondary network). Figure 2 also shows the primary user termination probability for N=3 presented in [3] and reproduced here.

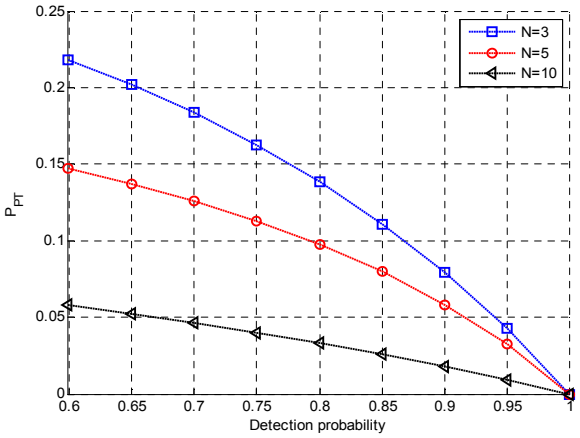


Figure 2. Primary user termination probability

2) Primary Network Capacity: Figure 3 shows that the capacity decreases when there is less accuracy on the detection of the primary presence. If the P_D is low, then there will be more interruptions and fewer calls will have the chance to regularly finish their transmission.

3) Secondary User Blocking probability: Figure 4 shows that, if the arrival rate of primary users' calls increases, then the secondary user blocking probability also increases. This occurs because there will be less transmission opportunity for the secondary network, resulting in more blocked secondary users' calls.

4) Probability that a secondary call is normally terminated: Figure 5 shows that, with the increase of the arrival rate of the primary network, secondary users' services will be interrupted more often. Therefore, the probability that a secondary user´s call successfully ends decrease. Figure 5 also shows the probability that a secondary call is normally terminated for different values of μ_2 and N=3 presented in [3] and reproduced here.

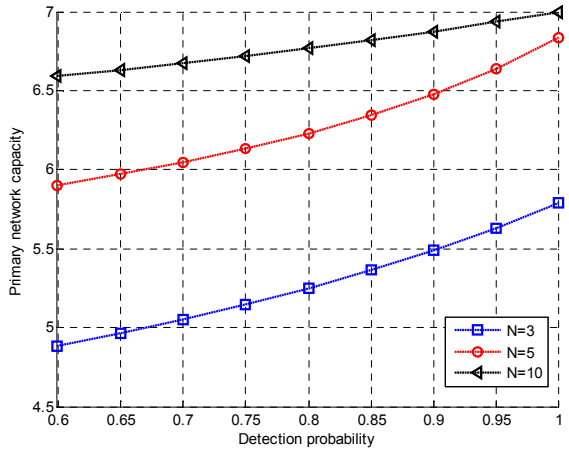


Figure 3. Primary network capacity

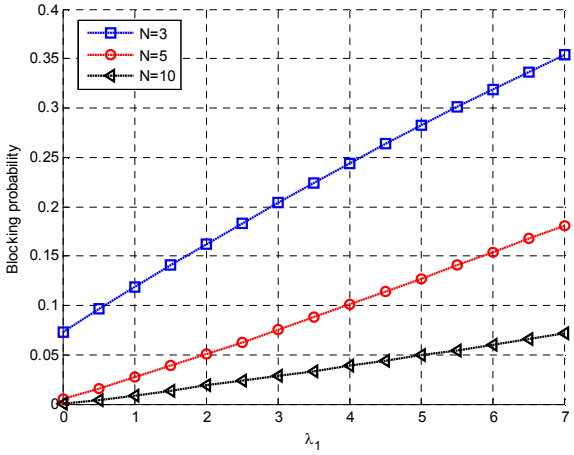


Figure 4. Secondary user blocking probability

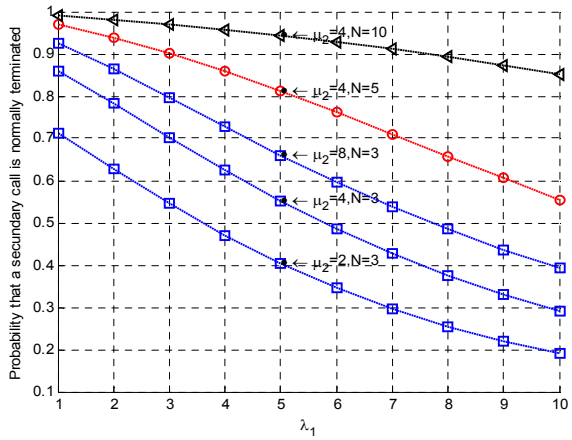


Figure 5. Probability that a secondary call is normally terminated

5) Secondary network Capacity: Figure 6 shows that low capacity is achieved if the arrival of primary users' calls are

frequent. Figure 6 also shows that higher capacity is achieved when more channels are available.

6) Spectrum Utilization: Figure 7 shows that better spectrum utilization can be obtained if there are less detections errors.

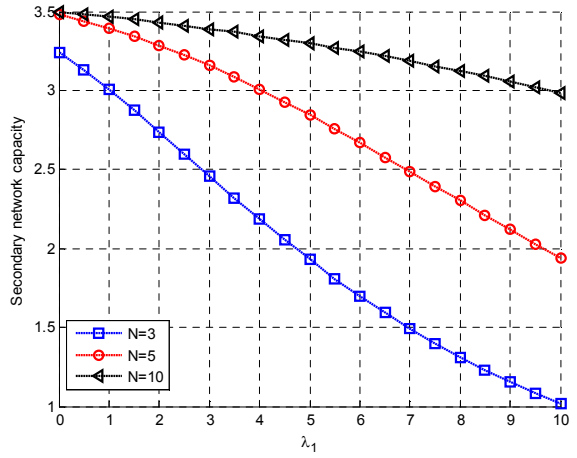


Figure 6. Secondary network capacity

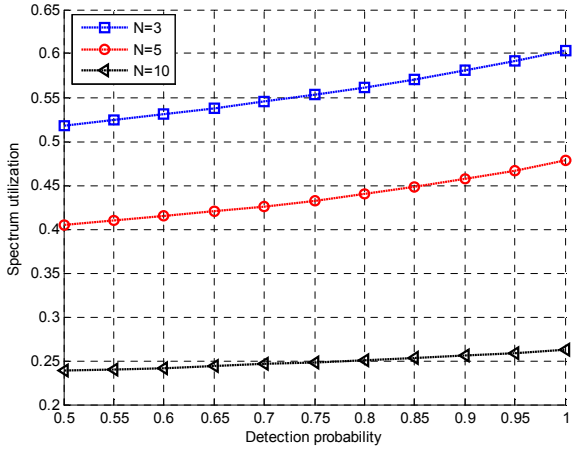


Figure 7. Spectrum utilization

In Figures 2-7, it can be seen that the performance of the opportunistic spectrum access gets better for larger number of channels N. It happens because if more channels are available, then less calls will be forced to terminate due to misdetections resulting in more capacity and better spectrum utilization.

VI. CONCLUSIONS

In this paper, an existing state diagram based approach for modeling the opportunistic access of a cognitive network with imperfect sensing is improved by allowing the state equations to be generalized to any numbers of channels. Additionally, we introduced four new parameters to evaluate the performance of such network. The developed analytical model allows us to see the effect of the imperfect sensing on the number of calls completed per time unit for both primary and secondary networks and the overall spectrum utilization for any number of channels in the system.

ACKNOWLEDGMENT

This work was partially supported by Finep, with resources from Funttel, Grant No. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicação) project of the National institute of telecommunications (Instituto Nacional de Telecomunicaçoes – Inatel), Bazil.

REFERENCES

[1] J. Mitola III, “Cognitive radio: An integrated agent architecture for software defined radio,” Ph.D. dissertation, Royal Institute of Technology, Sweden, 2000.

[2] E. W. M. Wong and C.H. Foh “Analysis of cognitive radio spectrum access with Finite user population”. IEEE Communication Letters, May 2009, Vol. 3, No. 5, pp. 294-296.

[3] I. Suliman, J. Lehtomaki, T. Braysy, and K. Umebayashi, “Analysis of cognitive radio networks with imperfect sensing”, IEEE 20th Symposium On Personal, Indoor and Mobile Radio Communications, 2009, pp. 1616-1620.

[4] T. M. N Ngatched, S. Dong, and A. S. Alfa, “Analysis of cognitive radio networks with channel aggregation and imperfect Sensing”, Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), 2011, pp 1 – 6.

[5] T. M. N Ngatched, S. Dong, A. S. Alfa, and J. Cai, “Performance analysis of cognitive radio networks with channel assembling and imperfect sensing”, IEEE International Conference on Communications (ICC), 2012, pp. 1688 – 1692.

[6] L. Jiao, F. Li, and V. Pla, “Dynamic channel aggregation strategies in cognitive radio networks with spectrum adaptation,” 2011 IEEE GLOBECOM, pp. 1-6.

[7] L. Jiao, F. Li, and V. Pla, “Modeling and performance analysis of channel assembling in multi-channel cognitive radio networks with spectrum adaptation,” IEEE Transaction on Vehicular Technology., July 2012, vol. 61, no. 6, pp. 2686–2697.

[8] I. Balapuwadege, L. Jiao, F. Li, and V. Pla, V. “Channel assembling with priority-based queues in cognitive radio networks: strategies and performance evaluation” IEEE transactions On wireless Communication, February 2014, Vol.13, No 2, pp. 630-645.

[9] S. Tang and B. L. Mark, “Modeling and analysis of opportunistic spectrum sharing with unreliable spectrum sensing” IEEE Trans. Wireless Commun., vol. 8, 2009, pp. 1934-1943.

[10] I. Akyildiz, B. Lo, and R. Balakrishnan, “Cooperative spectrum sensing in cognitive radio networks: A survey” Physical Communication, December 2010, pp. 40-62.

[11] L. Kleinrock, Queueing Systems, Volume I: Theory. Jonh Wiley and Sons, March 1975

 ISBN: 978-1-61208-450-3

Throughput Analysis in Cognitive Radio Networks Using Slotted Aloha Protocol with Imperfect Sensing

Pedro Ivo de Almeida Guimarães and José Marcos Câmara Brito
Institute National of Telecommunications - INATEL
Santa Rita do Sapucaí, Brazil
Email: pedroguimaraes11@gmail.com, brito@inatel.br

***Abstract*—This paper introduces an extension of a methodology for calculating the throughput in a cognitive radio network using the slotted aloha technique for medium access. This extension includes two new aspects: imperfect sensing and different transmission power levels in the secondary stations. The performance of the network is evaluated and numerical results are compared with the results for the original model. The new aspects considered in this paper make the model more realistic.**

Keyword— Cognitive Radio; Multiple Access; Throughput; Performance Analysis.

I. INTRODUCTION

During the last decades, emerging applications in wireless communication networks gained attention due to the increasing demand of high transmission rates services [1]. According to [1] and [2], the majority of the available spectrum frequencies has already been fixed and licensed to primary user's (PU's). A study undertaken by Spectrum Policy Task Force (STPF), linked to Federal Communications Commission (FCC), of United States of America (USA), has shown that in certain locations and during certain periods of time, some spectrum bands are widely used by PU's. However, in some other locations, there are many frequency bands that are partially occupied or not used at all [1]. According to the FCC, the usage of licensed spectrum bands varies between 15% and 85%. A way to overcome these limitations is to promote changes in the current licensing model, by allowing secondary users (SUs) to access the available spectrum, without causing harmful interference in the PU's communications. The efficiency in the usage of the radio spectrum would be improved in order to support the required demands [2]. Cognitive Radio (CR) is a technology that can be used to implement this approach.

CR is defined as a radio that can change its transmission parameters based on the environment in which it is operating. The main functions of cognitive radio include spectral detection, spectrum management, spectral mobility and spectrum sharing [2]. Its paramount objective is to provide adaptability to wireless transmission systems through Dynamic Spectrum Access (DSA) in order to optimize the performance and improve the use of the spectrum [1].

The components of a Cognitive Radio Network (CRN) Architecture can be classified in two groups: primary and secondary networks. The first is the licensed network infrastructure, which is entitled to exploit a certain band of the frequency spectrum. The secondary network is not licensed to operate in that band and their radio stations access the spectrum done in an opportunistic manner, exploring the unused bands by PU's, defined as a spectral holes.

Medium Access Control (MAC) is a key issue in CRN. In the primary network, the MAC protocols are important in order to organize the access to the channel of different PUs. In the secondary network, the MAC protocols have the responsibility to organize the access of SUs to the free channels of the primary network and prevent the licensed network from harmful interference. Several papers have analyzed the performance of MAC protocols in a CRN environment. Li et al. [3] analyzed the impact of imperfect sensing in a distributed multi-channel cognitive radio system. Moshksar et al. [4] analyzed the effect of power levels by SUs for achieving high rate per SU. Bayrakdar et al. [5] investigated the throughput of a slotted aloha-based random access CRN with capture effect in Rayleigh fading channels. In [6], the performance of the slotted aloha protocol, in both networks (primary and secondary), is also analyzed considering the capture effect and Rayleigh channels. The analyses presented in [5] and [6], however, do not consider the Packet Error Rate (PER) when two or more stations transmit simultaneously. This lack in the performance analysis has been solved by the extension published in [7]. However, the analyses presented in [6] and [7] do not consider two important aspects: imperfect sensing and different transmission power levels in SUs. The main goal of this paper is to extend the analysis presented in [6] and [7], considering these effects in the mathematical formulation.

The rest of this paper is organized as follows: in Section II, we present the model used in [6], and the extension presented in [7], called original model; the new model is proposed in Section III; numerical results and discussions are presented in Section IV; and conclusions are given in Section V.

II. SYSTEM MODEL

The analyzed architecture, which is illustrated in Figure 1, considers two networks (primary and secondary) that coexist in the same geographic region and frequency band. The secondary network contains a SAP and *Ns* users who compete for spectrum opportunities. The primary network has a primary access point (PAP) and *Np* PU's. According to [6] and [7], during a time slot, there are *Ip* PU's and *Js* SUs attempting to transmit their data packets to their respective access points. During a time slot, each PU or SU, which is not in the packet transmission state can generate a new packet with probability σ_p or σ_s respectively. Thus, the probability of no new PU or SU packet generation is $(1-\sigma_p)$ or $(1-\sigma_s)$, respectively. If a PU or an SU generates a data packet on its network, the packet is transmitted in the next time slot. If an error occurs during the transmission of the packet, it is then retransmitted with

 ISBN: 978-1-61208-450-3

probability σ_p or σ_s in the following time slot, until the packet is received successfully. If a PU or SU is in the retransmission state, no new packet can be generated.

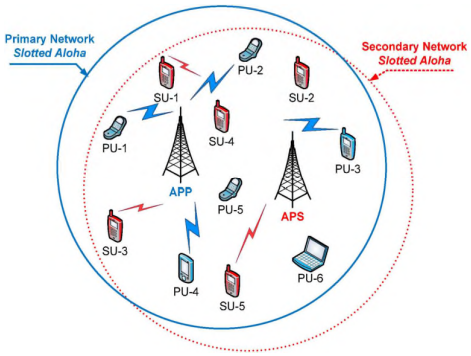


Figure 1. Primary and Secondary networks model using slotted aloha protocol

A. Fading Model for Primary and Secondary Network

In this paper, following [6] and [7], a quasi-static fading model is used, according to the Rayleigh statistical model, wherein the instantaneous power of the received signal has an exponential distribution, see (6).

B. Power Level Applied in the Network

Let X_p and X_s be the mean values of instantaneous power of the concerned packet from primary and secondary networks, respectively. Let Y and Z be the mean values of the interfering powers of one packet from primary and secondary network, respectively. In [6] and [7], we have $X_p = Y$ and $X_s = Z$. Keeping in mind that the SUs work with lower levels of transmission power, to minimize interference in the PUs, we define the following ratio:

$$\gamma_1 = \frac{X_p}{Z} = \frac{Y}{X_s}. \quad (1)$$

C. Analysis of the Capture Effect

In slotted aloha systems, packets that arrive at the receiver have different power levels due to the distance between the transceivers, the transmission power level, the channel shadowing and the signal fading. If the difference between the power levels of a concerned packet in relation to other interfering packets is higher than a threshold, called capture ratio (R), then the concerned packet can be detected by the receiver [8].

In [6] and [7], the authors consider a capture model where the interfering power is the sum of all received interfering powers from SUs and PU's. If the power of the concerned packet is higher than the interfering power and it satisfies the capture ratio R, then the receiver captures the concerned packet. The analysis presented in [6] considers that the captured packet can be detected by the PAP or SAP without errors and the analysis presented in [7] considers the effects of the PER. Assuming that the concerned packet is coming from a PU, then the capture probability ($P_{pcap \to PAP}$) can be calculated by [6]:

$$P_{pcap \to PAP}(I_p, J_s) = \Pr\left(\frac{x_p}{\sum_{i=1}^{I_p - 1} y_i + \sum_{j=1}^{J_s} z_j} > R \right), \quad (2)$$

where I_p and J_s are the respective numbers of PU's and SUs that can generate or retransmit a packet with probability σ_p or σ_s in a given time slot, x_p is the instantaneous power of the concerned packet generated by a PU, R is the capture ratio and y_i and z_j are the instantaneous powers of interfering packets generated in the primary and secondary networks, respectively [6].

If the concerned packet is coming from an SU, then the capture probability ($P_{scap \to SAP}$) is calculated as [6]:

$$P_{scap \to SAP}(I_p, J_s) = \Pr\left(\frac{x_s}{\sum_{i=1}^{I_p} y_i + \sum_{j=1}^{J_s - 1} z_j} > R \right), \quad (3)$$

where x_s is the instantaneous power of the concerned packet generated by an SU.

D. Packet Error Rate Analysis

The knowledge of the PER in communication systems is important, since most systems have data transmitted in packets [7]. Furthermore, the performance of the system is determined by PER instead of the Bit Error Rate (BER) or Symbol Error Rate (SER) [7][9]. The relation between PER and the system model is made by the Signal-to-Interference plus Noise Ratio (SNIR). According to [10], the Gaussian noise can be neglected in channels limited by the interference. Thus, in this paper, we consider the Signal-to-Interference Ratio (SIR).

In this article, following [7], the PER is calculated for a fading channel as a function of SIR, through the use of a fairly accurate upper bound for the considered system [9]. Knowing that $X_p = Y$ and $X_s = Z$ and applying (1), we obtain the mean value of SIR for the primary networks (Δ_p) and secondary networks (Δ_s), given by:

$$\Delta_p = \frac{1}{(I_p - 1) + J_s / \gamma_1}, \quad (4)$$

$$\Delta_s = \frac{1}{\gamma_1 I_p + J_s - 1}. \quad (5)$$

Let $f(\delta)$ be a function that relates the PER with the instantaneous SIR at the reception (δ), in an additive white Gaussian noise channel (AWGN), and $p(\delta)$ the probability density function of Signal-to-Interference Ratio (SIR) in the receiver, considering a Rayleigh channel, which has an exponential distribution given by:

$$p(\delta) = \frac{1}{\Delta} e^{-\delta/\Delta}. \quad (6)$$

According to [9], the PER represented by $P_{ave}(\Delta)$ can be calculated by the following expression:

$$P_{ave}(\Delta) = \int_0^{\infty} f(\delta) p(\delta) \, d\delta, \quad (7)$$

 ISBN: 978-1-61208-450-3

$$P_{ave}(\Delta) = \frac{1}{\Delta}\int_0^{\infty} f(\delta) e^{-\frac{\delta}{\Delta}} \, \mathrm{d}\delta.$$

Considering the modulation techniques, packet lengths and coding schemes, it is difficult to compute (7) for a general case. An approximation is then proposed for the upper bound for the PER, according to the following inequality [9]:

$$P_{ave}(\Delta) \cong 1 - \mathrm{e}^{-w_0/\Delta}. \quad (8)$$

The Packet Success Rate (PSR) is then given by the following equation:

$$PSR(\Delta) \cong \mathrm{e}^{-w_0/\Delta}, \quad (9)$$

where w_0 is a constant value for Rayleigh channel and its value can be computed by the following expression [9]:

$$w_0 = \int_0^{\infty} f(\delta) \, \mathrm{d}\delta. \quad (10)$$

Not considering channel coding and considering n-bit packets, $f(\delta)$ can then be obtained as follows [9]:

$$f(\delta) = \left\{1 - \left[1 - b(\delta)\right]^n\right\}, \quad (11)$$

where $b(\delta)$ is the BER in AWGN channels. Considering a BPSK modulation with coherent detection, $b(\delta)$ can be calculated by [8]:

$$b(\delta) = \frac{1}{2} erfc(\sqrt{\delta}). \quad (12)$$

Applying (12) in (11) and then (11) in (10), we can compute (using Mathcad software) w_0. Considering *n=127 bits* per packet, the same value used in [7], we obtain w_0*=3,4467*.

Faria et al. [7] consider that a packet received with error is discarded and will be retransmitted, following the MAC protocol, until it is successfully received.

E. Calculating the Total Throughput of the Original Model

The total throughput (S_{ot}) is defined as the total number of packets generated by PUs and SUs that are correctly received by the PAP and SAP, respectively, during a time slot. The primary throughput (S_{op}), secondary throughput (S_{os}) and total throughput S_{ot}, are given as [7]:

$$S_{op} = \sum_{i=0}^{N_p}\sum_{j=0}^{N_s}\binom{N_p}{i}\sigma_p{}^i(1-\sigma_p)^{N_p-i}\binom{N_s}{j}\sigma_s{}^j(1-\sigma_s)^{N_s-j} \cdot \left[(i)\left[\left(\frac{1}{R+1}\right)^{i-1} e^{-w_0(i-1)}\left(\frac{\gamma_1}{R+\gamma_1}\right)^{j} e^{-w_0\left(\frac{j}{\gamma_1}\right)}\right]\right], \quad (13)$$

$$S_{os} = \sum_{i=0}^{N_p}\sum_{j=0}^{N_s}\binom{N_p}{i}\sigma^i{}_p\left(1-\sigma_p\right)^{N_p-i}\binom{N_s}{j}\sigma^j{}_s\left(1-\sigma_s\right)^{N_s-j} \cdot \left[(j)\left[\left(\frac{1}{R\gamma_1+1}\right)^{i} .e^{-w_0(\gamma_1 I_p)}\left(\frac{1}{R+1}\right)^{j_s-1} .e^{-w_0(J_s-1)}\right]\right] \quad (14)$$

$$S_{ot} = S_{op} + S_{os}. \quad (15)$$

III. THE NEW MODEL

The expressions to compute the throughput presented in [7] do not consider imperfect sensing and different transmission power levels in the SUs. In this section, an extension of such model is presented taking into account these aspects, making the model more accurate.

We consider that there is an entity in the secondary network responsible for sensing and making decisions about the channel status. Depending on the decision made in a given time slotted, the SUs adapt their transmission power level in order to minimize the interference on the primary network.

A. Channel Sensing

The spectral sensing is one of the most critical parts of the CRN. We consider that a secondary user initially performs channel sensing, which can be modeled as a hypothesis testing problem. Various channel sensing methods, including energy detection, cycle stationary detection, and matched filtering, have been proposed and analyzed in the literature [11]-[14]. Regardless of which method is used, one common feature is that errors, in the form of missed detections and false-alarms, can occur.

We assumed that H_0 denotes the hypothesis that the primary users are inactive in the channel, and H_1 denotes the hypothesis that the primary users are active. Thus, $\hat{H}_0$ denotes the decision that the PU is absent in the channel and $\hat{H}_1$ denotes the decision that the PU is active in the channel. With the result of the decision and the true nature of the activity of the PU, we have four possible cases that are described below [12][13].

Case 1: correct detection probability, considering that PU is active

$$P_d = \Pr\left\{\hat{H}_1 \middle| H_1\right\}. \quad (16)$$

Case 2: false alarm probability, considering that PU is absent

$$P_f = \Pr\left\{\hat{H}_1 \middle| H_0\right\}. \quad (17)$$

Case 3: correct detection probability, considering that PU is absent

$$P_{nf} = \Pr\left\{\hat{H}_0 \middle| H_0\right\} = 1 - P_f. \quad (18)$$

Case 4: incorrect detection probability, considering that PU is active

$$P_{ed} = \Pr\left\{\hat{H}_0 \middle| H_1\right\} = 1 - P_d. \quad (19)$$

B. Power Level Applied in the Network

We considered that X_{sk} is the mean value of the instantaneous power of the concerned packet from the secondary network and Z_k is the mean value of the interfering power from one SU, where $X_{sk} = Z_k$.

In our notation, k denotes the decision about the state of the channel, where $k = 0$ represents that the decision is channel free and $k = 1$ that the decision is channel busy.

The transmission power of the SU when $k = 0$ is greater than when $k = 1$, with the goal to reduce interference in the primary network. Now, the ratio between the power of a packet from the primary network and the power of one from the secondary network depends on the decision about the channel state and is given by:

$$\gamma_k = \frac{X_P}{Z_k} = \frac{Y}{X_{sk}}. \tag{20}$$

C. Capture Effect for the New System

To compute the capture probability we consider that all SUs have the same decision about the state of the channel (free or busy). This approach is equivalent to considering that the decision process in the secondary network is centralized.

For the capture effect, if the concerned packet is generated by a PU, the probability of capturing this packet is given by (2), modified to consider that we have two different power levels in the secondary network:

$$\mathrm{P}_{\mathrm{pcap}_k \to \mathrm{PAP}}(I_p, J_s) = \Pr\left(\frac{x_p}{\sum_{i=1}^{I_p\text{-}1} y_i + \sum_{j=1}^{J_s} z_{jk}} > R \right), \tag{21}$$

If the concerned packet is generated by an SU, then the probability of capture is calculated modifying (3), resulting in:

$$\mathrm{P}_{\mathrm{scap}_k \to \mathrm{SAP}}(I_p, J_s) = \Pr\left(\frac{x_{sk}}{\sum_{i=1}^{I_p} y_i + \sum_{j=1}^{J_s - 1} z_{j_k}} > R \right), \tag{22}$$

where x_{sk} is the instantaneous power of the concerned packet generated by an SU.

In (21) and (22), x_s, x_p and y_i have exponential distribution (see (6)), $\sum z_k$ represents the instantaneous interfering power from the secondary network, that is obtained by adding J_s or J_s-1 independent identically distributed (iid) random variable with exponential distribution, which converges to an Erlang distribution, given by [15]:

$$f(\sum \mathrm{z}_k) = \frac{(C_k)^j (z_k)^{(j-1)} e^{-C_k z_k}}{(j-1)!}. \tag{23}$$

In (23), j represents the number of interfering users (j=Js for primary networks and j=Js-1 for secondary networks). The parameter C_k is associated to the mean value of the interfering power from one SU and is given by:

$$C_k = \frac{1}{Z_k} \tag{24}$$

Considering that all PUs or SUs are mutually independent, the joint probability density functions for the received powers, for the primary and secondary networks, are given, respectively, by:

$$f\left(x_p, y_i, \ldots, y_{I_p-1}, z_k\right) = \frac{1}{X_p} e^{-\frac{x_p}{X_p}} \prod_{i=1}^{I_p-1} \frac{1}{Y} e^{-\frac{y_i}{Y}} . \cdot \frac{(C_k)^j (z_k)^{(j-1)} e^{-C_k z_k}}{(j-1)!}, \tag{25}$$

$$f\left(x_{sk}, y_1, \ldots, y_{I_p}, z_k\right) = \frac{1}{X_{sk}} e^{-\frac{x_{sk}}{X_{sk}}} \prod_{i=1}^{I_p} \frac{1}{Y} e^{-\frac{y_i}{Y}} . \cdot \frac{(C_k)^j (z_k)^{(j-1)} e^{-C_k z_k}}{(j-1)!}. \tag{26}$$

Now, to compute the capture probability we need to solve the integrals given by (27) and (28), respectively for primary and secondary networks.

$$\int_0^\infty \ldots \int_0^\infty \int_{R\left(\sum_{i=1}^{I_p-1} y_i + z_k\right)}^\infty \frac{1}{X_p} e^{-\frac{x_p}{X_p}} \prod_{i=1}^{I_p-1} \frac{1}{Y} e^{-\frac{y_i}{Y}} . \cdot \frac{(C_k)^j (z_k)^{(j-1)} e^{-C_k z_k}}{(j-1)!} dx_p dy_i \ldots dy_{I_p-1} dz_k, \tag{27}$$

$$\int_0^\infty \ldots \int_0^\infty \int_{R\left(\sum_{i=1}^{I_p} y_i + z_k\right)}^\infty \frac{1}{X_s} e^{-\frac{x_s}{X_s}} \prod_{i=1}^{I_p} \frac{1}{Y} e^{-\frac{y_i}{Y}} . \cdot \frac{(C_k)^j (z_k)^{(j-1)} e^{-C_k z_k}}{(j-1)!} dx_s dy_i \ldots dy_{I_p} dz_k. \tag{28}$$

Now we consider the value of γ_k depending on the channel status decision. In this context, the decision can result in four values, as described below:

- P_d, if the channel is busy and the decision is busy, we have: k=1 and $\gamma_k = \gamma_1$;
- P_{ed}, if the channel is busy and the decision is free, we have: k=0 and $\gamma_k = \gamma_0$;
- P_f, if the channel is free and the decision is busy, we have: k=1 and $\gamma_k = \gamma_1$;
- P_{nf}, if the channel is free and the decision is free, we have: k=0 and $\gamma_k = \gamma_0$.

Thus, γ_1 represents the ratio (see (20)) when the decision about the channel is busy and γ_0 represents the ratio when the decision about the state of the channel is idle.

Now, we have the capture probability in the primary network, considering that PU is always active and the result of the decision can be right (P_d) or wrong (P_{ed}), as given below:

$$\mathrm{P}_{\mathrm{pcap}\to\mathrm{PAP}}(I_p,J_s)=\Pr\left[\left(\left(\frac{\gamma_1}{R+\gamma_1}\right)^{J_s}P_d+\left(\frac{\gamma_0}{R+\gamma_0}\right)^{J_s}P_{ed}\right).\left(\frac{1}{R+1}\right)^{I_p-1}\right], \quad (29)$$

For the secondary network, we have the capture probability as given below:

$$\mathrm{P}_{\mathrm{pcap}\to\mathrm{SAP}}(I_p,J_s)=\Pr\left(\left(\left(\frac{1}{R\gamma_1+1}\right)^{I_p}P_d+\left(\frac{1}{R\gamma_0+1}\right)^{I_p}P_{ed}\right).\left(\frac{1}{R+1}\right)^{J_s-1}\right) \quad (30)$$

D. Packet Error Rate

To compute the PER, we use the same approach as in Section II. However, now the SIR depends on the transmission power levels, which depends on the channel state decision. Thus, we have:

$$\Delta_{p_k}=\frac{1}{\left(I_p-1\right)+\frac{J_s}{\gamma_k}}, \quad (31)$$

$$\Delta_{sk}=\frac{1}{\gamma_k I_p+\left(J_s-1\right)}. \quad (32)$$

The PER and PSR are computed for k=0 and k=1, using expressions similar to (8) and (9).

E. Total Throughput of the New Model

The total throughput of the network in the new model is defined as the total number of packets generated by PUs and SUs that are correctly received by the PAP and SAP, respectively, during a time slot.

Considering only correctly received packets, the primary network throughput, represented by (S_{np}), the secondary network throughput, represented by (S_{ns}), and the total throughput of the network, represented, by (S_{nt}), are given, respectively, by (33), (34) and (35).

$$Snp=\sum_{i=0}^{Np}\sum_{j=0}^{Ns}\binom{Np}{i}\sigma_p{}^{i}\left(1-\sigma_p\right)^{Np-i}\binom{Ns}{j}\sigma_s{}^{j}\left(1-\sigma_s\right)^{Ns-j}.\left(\left[\left[\frac{\gamma_0}{R+\gamma_0}\right]^{j}e^{-w_0\left(\frac{j}{\gamma_0}\right)}P_{ed}+\left[\frac{\gamma_1}{R+\gamma_1}\right]^{j}e^{-w_0\left(\frac{j}{\gamma_1}\right)}P_d\right].\left[(i)\left[\frac{1}{R+1}\right]^{i-1}e^{-w_0(i-1)}\right]\right) \quad (33)$$

$$Sns=\left(\sum_{i=0}^{Np}\sum_{j=0}^{Ns}\binom{Np}{i}\sigma_p{}^{i}\left(1-\sigma_p\right)^{Np-i}\binom{Ns}{j}\sigma_s{}^{j}\left(1-\sigma_s\right)^{Ns-j}(j).\left(\left[\left[\frac{1}{R\gamma_0+1}\right]^{i}e^{-w_0(\gamma_0 i)}P_{ed}+\left[\frac{1}{R\gamma_1+1}\right]^{i}e^{-w_0(\gamma_1 i)}P_d\right].\left[\frac{1}{R+1}\right]^{j-1}e^{-w_0(j-1)}\right)\right) \quad (34)$$

$$S_{nt}=S_{np}+S_{ns}. \quad (35)$$

Based on (33) and (34) we can observe that the false alarm probability does not interfere with the throughput of the network.

IV. NUMERICAL RESULTS

Let m be a relation among each network transmission probabilities [6][7]:

$$m=\frac{\sigma_s}{\sigma_p}. \quad (36)$$

The curves presented in Figures 2, 3 and 4, show the throughput for the primary network, secondary network and total throughput. They are plotted considering: the original model introduced in [7] and the new model proposed here. We take into account imperfect sensing and different transmission power levels regarding the channel's state decision. To compare the numerical results between the new model proposed and the original model, we considered the same parameters used in [7] i.e., Np= Ns=30, R=3dB, $\gamma = 10$, and m alternating between 1, 2 and 5. For the new model, we set $P_d = 0.8$, $\gamma_0 = 5$, and $\gamma_1 = 10$. Observing Figures 2, 3 and 4, for the P_d value assumed in this paper, we can conclude that:

- The throughput in the primary network decreases when we consider the effects of the imperfect sensing.
- Regarding the secondary network, the throughput is not influenced by the imperfect sensing.

As future work, it is important to investigate the influence of the parameters P_d, γ_o, γ_1, Np, and Ns when imperfect sensing is considered.

Additionally, as future work, one can investigate the influence of channel coding techniques as a solution to increase the throughput of cognitive networks.

Also, as a future study, one can investigate adaptive modulation schemes and channel coding techniques as solutions to increase the throughput in the networks.

 ISBN: 978-1-61208-450-3

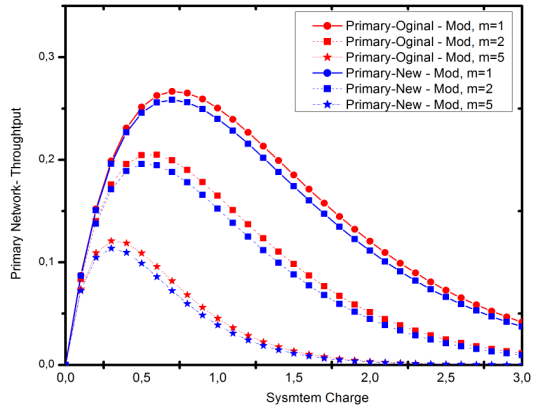


Figure 2. Comparison between original and proposed normalized primary throughput for both models with $Np = Ns = 30$, $\gamma_0 = 5$, $\gamma_1 = 10$, $R = 3dB$, and $P_d = 0.8$.

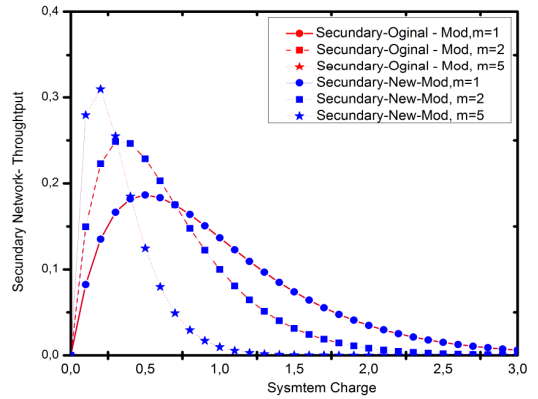


Figure 3. Comparison between original and proposed normalized Secondary throughput for both models with $Np = Ns = 30$, $\gamma_0 = 5$, $\gamma_1 = 10$, $R = 3dB$, and $P_d = 0.8$.

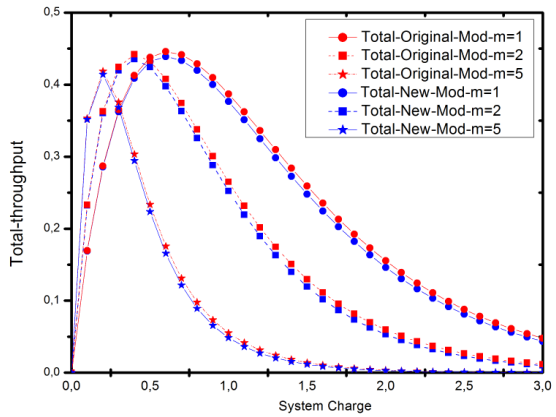


Figure 4. Comparison between original and proposed normalized Total throughput for both models with $Np = Ns = 30$, $\gamma_0 = 5$, $\gamma_1 = 10$, $R = 3dB$, and $P_d = 0.8$.

V. CONCLUSIONS

In this paper, we extended the analysis presented in [7], considering the effect of imperfect sensing in the throughput of a cognitive radio network that employs slotted aloha multiple access protocol in the primary and secondary networks. Also, we consider different transmission power levels in the secondary network as a function of the decision in this network about the state of the channel.

We observed that imperfect sensing and different transmission power levels reduce the throughput in primary network and have no influence in the secondary network.

ACKNOWLEDGMENT

This work was partially supported by Finep, with resources from Funttel, Grant No. 01.14.0231.00, under the Radiocommunication Reference Center (*Centro de Referência em Radiocomunicações* - CRR) project of the National Institute of Telecommunications (*Instituto Nacional de Telecomunicações* - Inatel), Brazil.

REFERENCES

[1] F. Akyildiz, W. Lee, M. C. Vuran, and S. Mohanty, "Next Generation/Dynamic Spectrum Access/ Cognitive Radio Wireless Networks: A Survey," Computer Networks, Elsevier, vol. 50, May 2006, pp. 2127- 2159.

[2] E. Hossain, D. Niyato, Z .Han, "Introduction to Cognitive Radio Dynamic Spectrum Access and Management in Cognitive Radio network", New York: Cambridge University Press, P.39-73. 2009.

[3] X. Li, Q. Song, J. Wang, H. Tao and J. Zhang, "Improved Idle Channel Utilization in Distributed Multi-channel Cognitive Radio Systems". IEEE 77th Vehicular Technology Conference (VTC Spring), 2013, pp. 1 - 6.

[4] K. Moshksar, A. K. Khandani, "Randomized Masking in Cognitive Radio Networks," IEEE Transactions On Communications, Vol. 61, No. 7, July 2013.

[5] M. E. Bayrakdar, S. Atmaca , A. Karahan, A lotted Aloha-Based Random Access Cognitive Radio Network With Capture Effect In Rayleigh Fading Channels," International Conference on Electronics, Computer and Computation (ICECCO), 2013, pp. 72 - 75

[6] Z. Yang, "Investigations of Multiple Access Protocols in Cognitive Radio Networks", Ph.D. dissertation, Dept. of Elect. and Computer Engineering., Stevens Inst. of Technology, Hoboken, NJ. 2010.

[7] A. Faria, and J. M. C. Brito, "Analyze of throughput in cognitive radio networks using Slotted Aloha", SBrt. vol.12, September 2012, pp.13-16.

[8] Y. Onozato, J. Liu, and S. Noguchi, "Stability of a Slotted Aloha system with Capture Effect," IEEE Transactions. on Vehicular Technology, vol. 38, February 1989, pp.31-36.

[9] Y. Xi, A. Burr, J. Wei and D. Grace, "A General Upper Bound to Evaluate Packet Error Rate over Quasi-Static Fading Channels". IEEE Transactions on Wireless Communication, vol. 10, May 2011, pp. 1373-1377.

[10] S. Kucera, S. Aissa, and S. Yoshida, "Adaptive channel allocation for enabling target SINR achievability in power-controlled wireless networks", IEEE Trans. on Wireless Communication, vol. 9, February 2010, pp. 833-843.

[11] M. A. Abdulsattar, and Z. A. Hussein, "Energy Detection Technique for Spectrum Sensing in Cognitive Radio: A Survey". International Journal of Computer Networks & Communications (IJCNC) vol. 4, September 2012.

[12] G. Ozcan, M. C. Gursoy, and S. Gezici, "Error Rate Analysis of Cognitive Radio Transmissions with Imperfect Channel Sensing," IEEE Transactions on Wireless Communication vol. 13, March 2014, pp.1642-1655.

[13] D. S. Chaves, "Analysis of Spectral Sensing Power Detection", dissertation Master's dissertation in Electrical Engineering, Publishing PPGEE.DM-502/2012, Dep. of Engineering. Electrical. Univer. Of Brazilian, pp. 103, 2012.

[14] T. Yucek, and H. Arslan, "A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications" IEEE Communications Survey & Tutorials, vol. 11, February, 2009, pp. 116-130.

[15] C. Ibe Oliver, Fundamentals of Applied Probability and Random Processes, Elsevier Academic Press, pp. 136-139, 2005.

 ISBN: 978-1-61208-450-3

Reliablity, Resiliency and Fault Management in Network Function Virtualization

Ramachandran Sathyanarayanan
HCL Technologies LTD
Noida, India
email: sathyanarayananr@hcl.com

***Abstract*— Network Function Virtualization (NFV) will change the way Telecom Service providers (TSPs) have been using network functions on the proprietary hardware for various telecom and networking services. Virtualization of network functions has provided many benefits to Telecom Service Providers (TSP). While NFV has bought numerous benefits to the TSP in terms of speed of deployment, flexibility and cost reduction, the additional virtual layers introduced directly impacts the reliability, resiliency and fault management. These include, but are not limited to, Latency between Virtual Machine (VM), Latency between Network Function Virtualization Infrastructure (NFVI) nodes, Network Interface Card (NIC) availability, and Processor availability. This paper outlines and summarizes the challenges regarding reliability, resiliency and fault management in the carrier-grade NFV environment and discusses various models and features of reliability, resiliency and fault management needed to deploy a Virtual Network Function (VNF) in the service provider environment.**

***Keywords-NFV-reliability; service availability*.**

I. INTRODUCTION

This paper aims at discussing, summarizing and recommending various reliability and fault management techniques that are being used across various NFV [1] solutions in the industry. NFV makes the network flexible, agile and programmable. The flexibility and programmability are the biggest benefit of virtualizing network functions. However, virtualization introduces challenges to ***reliability*** and fault management mechanisms. Commodity switches and servers are prone to faults and failures. Hence, ***reliability*** and fault management are built into software, and not into hardware, in the virtualization world. ***Reliability*** and fault management are more challenging in software than in hardware. This paper discusses, summarizes and collates various ***reliability*** and fault management techniques that will be required in the NFV carrier-grade solution. These techniques will need to be implemented in all the components of the Network Function Virtualization Architectural framework (Figure 1).

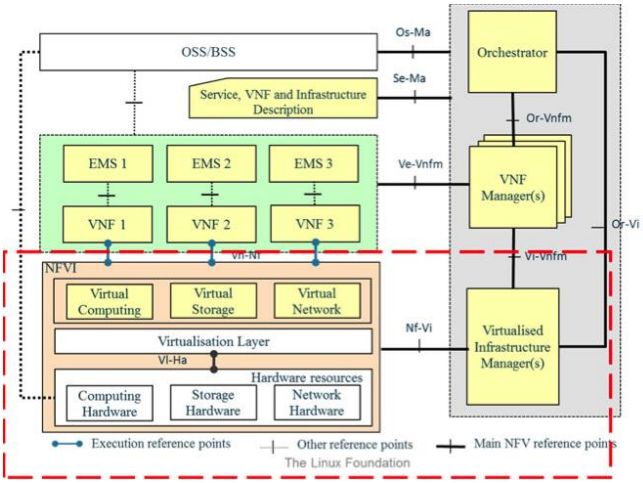


Figure 1. NFV Architectural Framework [1].

Figure 1 shows the NFV architectural framework depicting the functional block and reference points [1]. Solid lines depict NFV specific reference points and the dotted line depicts existing reference points in the operator network.

The NFVI block contains the infrastructure functional block including physical hardware devices, virtualization layer and virtual devices. VNFs are hosted on the NFVI block. The Element Management System (EMS) does the management function for one or more VNFs. NFV Management and Orchestration layer has management components managing Virtual infrastructure (VIM), VNF Manager and Orchestrator. Existing Operations and Business support systems (OSS/BSS) of the operator will interface with the NFV framework.

Network Function Virtualization Architectural framework focuses on changes what will happen to the operator network due to network virtualization. New functional blocks and reference points are introduced in the operators' network. NFV framework emphasizes the fact that the exact Virtual Network Function (VNF) deployment location may not be visible. VNF can be deployed in any of the physical resources across a geographic location as long as end-to-end service quality levels are met. This paper summarizes and collates all methods that would be needed for reliability and fault management in NFV.

 ISBN: 978-1-61208-450-3

The rest of the paper is structured as follows. Section II and Section III describe service availability techniques of VNF and Management and Orchestration (MANO), respectively. Section IV and Section V discuss Fault management and Fault prediction techniques. Section VI concludes the paper.

II. SERVICE AVAILABILITY OF VNF

NFV provides better service flexibility and scalability to operators than the network functions on the proprietary hardware. However, the operators must ensure that their NFV platform provides the same dependable carrier-class reliability as expected by the customers. Carrier grade reliability ensures network uptime and availability of 99.9999% [3]. This means, the network should not be down for more than 32 seconds in a year. To ensure six-nine level of reliability, all the blocks of the NFV architecture should be designed as below.

A. Service continuity

During an overload condition or during fault, VNF can be migrated to another server/data center. This helps in higher resiliency of the infrastructure. There can be two types of VNF based on ease of migration in the field. For certain services, VNF can be migrated stateless, e.g., Domain Network Service (DNS). For other services, VNF state-information needs to be maintained, e.g., virtual IP-based Multimedia service (vIMS). During migration of services that need to maintain their state, it is important to first know that the node to which service is being migrated has enough infrastructure capabilities (compute, storage) before migration [2].

Microsoft's Live Migration [4] and VMware vMotion [5] feature used shared storage for the live migration of the VMs from one host to another. Later advances of these features supported "shared nothing" migration. This has enabled long distance migration of VMs across data centers. A most recent trend is the ability to do container-based live migration across data centers of different cloud services (e.g., between Amazon webservices and Microsoft Azure). Openstack recommends multiple block and object storage for redundancy [6].

B. Network topology transparency

Like any resilient system, it is important to have redundancy. In the virtualized environment, it is desirable and it is possible to have the standby node in a different topological and geographically sites [2]. During fault leading to migration of VM, additional configurations must be avoided and the transition should be seamless. Virtualization of networks has helped in achieving this. Software Defined Network technologies like VMware NSX [7] has virtualized the networks and Virtual Extensible LAN (VxLAN) overlay network has helped in making network topology transparent.

C. Network function priority

When a hardware fault occurs, all the VNFs running on the hardware need to be moved. However, it is possible that resources may not be available for all the VNFs. In such a scenario, high priority VNF should be moved to other server/location first. Hence, the priority of the VNF should be identified before the actual occurrence. The Virtual Infrastructure Manager (VIM) is responsible for identifying priorities and making decisions on relocation and suspension as it maintains overall resource usage of Virtual machines [2].

Service availability can be at different levels. Voice service and real-time services will have higher service level than online data services. Currently, ITU Y 2171 [8] and ITU Y 2172 [9] define priority levels for call admission control and restoration of service in next generation networks. The same will be applicable in the NFV infrastructure as well.

1) Level 1: Control plane traffic

This receives the highest level priority. Traffic includes control services essential for network operation and emergency telecommunication services. If more bandwidth is available, other traffic can be given this priority.

2) Level 2: Real-time services

Traffic with this priority level will receive lower restoration assurance compared to priority level 1. This includes voice/video for which real-time data flow happens.

3) Level 3: Data services

Transaction and message service fall in this category. Virtual Private Network (VPN), Short Message Service (SMS) are few examples in this category.

4) Level 4 : Internet Service Providers (ISP) services

Traffic in this priority receives the least assurance in service restoration. Tradition ISP services like e-mail and Web traffic are examples for these services. High availability feature helps in giving priority and attachment of the VM to a particular host.

D. VNF Replication:

In case the VNF gets overloaded due to a peak service request, it is desirable to replicate the VNF and run it in another host instead of migrating the VNF to another host with higher performance parameters. Load balancing of the services could be done between parent and child VNF. The current services should not be affected, but the new services can be processed by the new instance of the VNF. VIM will be responsible for the load balancing and optimizing the service.

Most of the NFV solutions provide this feature. VIM first identifies the service overload and the additional VM will be spawned based on the overload conditions.

 ISBN: 978-1-61208-450-3

III. SERVICE AVAILABILITY OF MANO

NFV MANO is involved in managing the orchestration of the infrastructure and VNF management. Hence, it is important for NFV MANO to be highly reliable. It should prevent overloading of VNF, dynamic load adaptation and quick service creation. Failure of any VNF MANO components should be isolated and it should not impact VNF services. High availability techniques need to apply for NFV MANO components to minimize the failures, and, even a failure occurs, a mechanism should be in place to bring it up rapidly. NFV Mano will detect, analyze and correlate events in the infrastructure and VNF; it should cause recovery in a timely manner. NFV MANO will have capability to detect overload conditions and load balance across VMs. NFV MANO should also deploy VMs accordingly, so that time taken to recovery will be faster. If state information is maintained in any cluster of hardware for a VM, it is important to have VM running in same cluster and in a cluster from which retrieving and bringing back the VM; this will take less time. For example, VMware MANO components, vCenter and vCloud director (vCD) have redundancy features built into them. Multiple instances of vCenter and vCD cells are hosted on different servers/clusters to provide continuous service availability. VMware uses shared storage to store state information, which can be retrieved within the same cluster to restart a VM.

IV. FAULT MANAGEMENT

A fault is a flaw in the system that causes error; it could be a unforeseen fault, like a software bug or a known fault that could occur due to system limitation, like servers/CPU/RAM or hard disk. An active fault will be observed in the system like alarms, error messages, service unavailability, or a service outage. Faults could be internal or external. An internal fault is caused within the system, such as software bug or it could be triggered externally, like following events such as [2]:

1) Cyber attacks on VNF, MANO, VIM or Orchestrator.

2) Large scale disaster destroying the hardware.

3) Enviornment challenges – Increase in temperature/interference.

4) High traffic impacting the service – like special event broadcast.

5) Failure of isolated device/software – Like NIC failure throttling incoming/outgoing traffic.

6) Accidents/Mistakes – e.g., wrong configuration causing high traffic on a VNF instances causing overload traffic.

A fault management system generally comprises of detecting the challenges in the system, preventing faults in the system and restoring services. A fault management system will have a set of mechanisms which will reduce the probability of failure and reduce the impact when failure occurs. It could be like firewall rules which actively block malicious traffic or a system which has redundancy, measuring the metrics, identifying signs of fault occurrence and having a redundancy mechanism in place to prevent service outage.

If a fault happens in the system, the first remediation is done if possible and then restoration is done. Remediation is containing or lessening the impact of the damage. For example, if the resource is constrained in the NFV environment for the vIMS video call, the call can be first downgraded to voice call, lessening the impact, and, once resources are available, it can be converted to video call again. Remediation is not always possible. If a link goes down between two data centers, either a full restoration can be done via a redundant link or a system outage occurs, if redundant links are not available.

Here are some examples of faults that are introduced by virtualization and approaches how they can be detected in each NFV layer:

1) Hardware failure NFV Infrastrcuture: It can be detected via Memory or Bus errors. Event notification can be sent to management layer for handling for remediation or recovery.

2) Virtualized infrastructure management (VIM): Heartbeat messages are sent to each host from VIM. Any missing response can be treated as host/hardware failure. Notification and handling can be sent to a Manager for remediation or recovery measures and for migrating VMs on the host to another host.

3) NFV orchestrator: VIM outage notification. VIM should inform NFV orchestrator during outage (proacvtive) or NFV orchestor can implement heartbeat messages to VIM during notification. Notification and handling can be sent to VNF manager for remediation or restoration procedure.

4) VNF: Any VNF running on faulty hardware will experience a abstract device error, like ping failure or software crashes or error log generation. Notification and handling message can be sent to VNF Manager for remediation or restoration procedure.

5) Hypervisor, host OS and guest OS failures:

a) VIM: Periodic monitoring of the hypervisor, guest OS and host OS. Notification and handling can be done by VIM for reboot of the faulty OS.

b) VNF: Any VNF running on faulty OS will experience a abstract device error, like ping failure or software crashes or error log generation. Notification and handling message can be sent to VNF Manager for remediation or restoration procedure.

6) Migration Failures

a) NFV Infrastructure : Failed migration can be detected at NFVi as incomplete migration, errors in storage etc. Notification and handling can be sent to VIM for restoration procedures.

 ISBN: 978-1-61208-450-3

b) VIM: Failed migration or incorrect suspend or restoration procedure can be detected at hypervisor which will have the knowledge of failure. VIM can do Notification and resotoration procedures.

c) VNF: Failed migration or incorrect suspend or restoration procedure can be detected at VNF which will have the knowledge of failure. Notifications can be sent to a VNF manager for restoration procedures.

7) Scaling and Descaling errors

a) VIM: Scaling and Descaling errors, failure to scale or descale on overload condition can be detected at VIM as any infrastructure like CPU/NIC would detect overload condition. VIM can initiate notification and restoration procedures..

b) VNF Manager: Scaling and Descaling errors, failure to scale or descale on overload condition can be detected at VNF manager as VNF will show application specific errors (call drops, traffic throttle, etc.). VNF manager can initiate notification and restoration procedures.

The examples above were failures introduced by virtualization. At each level of NFVi, a failure can be predicated for failure prevention, containment and overload conditions.

V. FAILURE PREDICATION, PREVENTION AND CONTAINMENT

Failure prevention is a measure of error avoidance during system planning, design or deployment and also avoiding failures once the system is operational [2]. Error avoidance during system planning, design or deployment involves quality assurance measures, design reviews, testing, and quality control procedures. Error avoidance during system operation involves fault monitoring, fault predication and fault control. Fault monitoring involves log collection, data analysis and possible fault predication [10]. Diagnostic analysis is then triggered for the confidence level of the fault by deep analysis and trend monitoring. Once fault is predicted, fault control procedures will be triggered [2]:

A. Failure Prediction

Failure model and frequency in the NFV environment are different from legacy TSP environment because of following reasons:

1) Integration of open source software with the generic hardware.

2) System complexity due to additional virtual components.

3) Dyanamic nature of the virtualization – migration, elasticity, upgrades and reconfigurations

4) Administrator/user inexperiance with virtualization

Failure prediction and monitoring the health of the system needs monitoring of real-time resource usage, such as CPU usage, memory usage, network and IO usage, or its loss rate. Sensitivity analysis of the one or more of the above parameters can be performed for failure prediction system. Trend identification and data correlation analysis can be done to understand system health. Advanced statistical analysis (Null hypothesis) [12] can be implemented in the failure predication system to determine the probability of failure based on the past data and the current trend. When certain hardware or software module is being deducted for possible failure, deep diagnostic analysis like hardware diagnostics should be done for failure deduction. Each hardware and software module will provide interface to management module for acquiring run-time and performance state information. Log analysis is another input for failure predication framework. Any error by hardware or software module deducted will be accompanied by the severity of the issue as well. Failure predication framework is located in the centralized module (orchestrator or external entity as it requires log analysis, fault correlation, correlation analysis from different layers).

B. Failure Containment

Failure containment is preventing failure propagation to different components [11]. It is done by

1) Isolating the failed system.

2) Propogation failure information to other components and components initiating redundant or back-up procedures.

In an NFV environment, this is done by having redundant VNF. Based on the resource availability, VIM should assign independent resources to a VM. Hence, failure of a resource or VM will be self-contained within a VM. VM along with hypervisor can be considered as independent entity for containment.

C. Overload Prevention

System overload can cause issues like latency, resources overrun, and memory leaks. In virtual environment in addition to load on the guest OS, load on the hypervisor will also become a factor. Hence, any overload control mechanism should take into account hypervisor load as well. NFV environment can use elastic resource management when load on the VNF increases gradually. In telco environment, traffic can increase significantly in a short period of time. Elastic resource management cannot be used in this scenario as the physical resources are already overloaded. In such cases, traditional overload prevention mechanism like call admission control could be used.

D. Prevention of single point of failure

Single point of failure should be avoided at any point of the NFVI framework – within VNF, between VNFs, and hypervisors.

 ISBN: 978-1-61208-450-3

1) VNF component running the same functionality should be deployed with appropriate anti-affinity rule to avoid single point of failure.

2) For disaster recovery, VNF could be deployed in a different geographical location.

3) VNF should have alternate network path for access in case of a link failure.

VI. CONCLUSION

In the virtual environment, to achieve carrier grade **reliability,** failure prediction, failure prevention, containment, and fault management must be implemented. Redundancy, high availability, migration of VNF is to be done to achieve continous service availability. Various techniques discussed should also be deployed in all components of the NFV Framework – VNF, VIM, and MANO.

REFERENCES

[1] ETSI GS NFV 002 – V1.1.1 (2013-10) – Network Function Virtualization – Architectural Framework

[2] ETSI GS NFV – REL 001 v1.1.1 (2015-01) – NFV: Resiliency requirement

[3] http://electronicdesign.com/communications/carrier-grade-reliability-must-have-nfv-success

[4] https://technet.microsoft.com/en-in/library/hh831435.aspx

[5] http://www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf

[6] http://docs.openstack.org/arch-design/storage-focus-operational-considerations.html#fault-tolerance-and-availability

[7] https://www.vmware.com/files/pdf/products/nsx/vmw-nsx-network-virtualization-design-guide.pdf

[8] ITU-T Rec. Y.2171 Admission control prioity levels in Next Generation Networks

[9] ITU-T Rec. Y.2172 Service restoration priority levels in Next Generation Networks

[10] https://wiki.opnfv.org/_media/doctor/doctorfaultmanagementandmaintenance.pdf

[11] Z. Amin, N. Sethi, and H. Singh, “Review on Fault Tolerance Techniques in Cloud Computing” in The International Journal of Computer Application (0975-8887), vol. 116, no. 18, April 2015

[12] J. Neyman, and E.S. Pearson, (January 1, 1933). "On the Problem of the most Efficient Tests of Statistical Hypotheses". *Philosophical Transactions of the Royal Society A* **231** (694–706): 289–337.

 ISBN: 978-1-61208-450-3

A STRIDE-based Security Architecture for Software-Defined Networking

Fabian Ruffy

Ludwig-Maximilians-Universität München
Communication Systems and Systems Programming
Munich, Germany
Email: ruffy@cip.ifi.lmu.de

Wolfgang Hommel, Felix von Eye

Leibniz Supercomputing Centre
Munich Network Management Team (MNM-Team)
Garching b. München, Germany
Email: [hommel,voneye]@lrz.de

***Abstract*—While the novelty of Software-Defined Networking (SDN) — the separation of network control and data planes — is appealing and simple enough to foster massive vendor support, the resulting impact on the security of communication networks infrastructures and their management may be tremendous. The paradigm change affects the entire networking architecture. It involves new IP-based management communication protocols, and introduces newly engineered, potentially immature and vulnerable implementations in both network components and SDN controllers. In this paper, the well-known STRIDE threat model is applied to the generic SDN concepts as a basis for the design of a secure SDN architecture. The key elements are presented in detail along with a discussion of potentially fundamental security flaws in the current SDN concepts.**

Keywords–Software-Defined Networking; STRIDE; Security Architecture; Network Security; Security Analysis.

I. INTRODUCTION

The Internet, and with it the use of IP-based protocols, has grown far beyond the expectations of its inventors more than 30 years ago. Hundreds of improvements to data transport, routing, and management protocols have been suggested and implemented since. Software-Defined Networking (SDN), however, may very well be the paradigm shift that will have the most important impact so far on how communication networks are provisioned and operated in the future. SDN's core idea is decoupling the data and control plane of network components and moving the control plane functionality to commonly used, separate SDN controllers. This concept is as ingenious as it seems simple. Nevertheless, it is inherently tied to fundamental changes regarding network management.

As it can be observed with promising new technologies, vendors frequently attempt to gain an advantage over potential rivals and promote the innovative functionality of their new products along with their supposed benefits. With this mindset and time-to-market constraints in focus, important real-world requirements such as mixed environments for smooth transitions and information security aspects are commonly treated as afterthoughts. Regarding SDN, it is fair to say that most setups have been installed and tested in lab environments. Several real-world data centre deployments are known, and an important research area is the application of SDN concepts to Internet-scale backbone infrastructures. This potential application scope gives security in SDN the utmost importance.

Despite several established alternatives, the OpenFlow protocol has become the de facto standard interface between SDN-based control and data planes. The standard committee maintaining the protocol is the Open Networking Foundation (ONF), which also addresses potential vulnerabilities in publications. However, security in SDN has been largely neglected, which can be derived from the lack of mutual interoperability of many OpenFlow implementations: Functionality and interoperability are significant selling points for vendors, and are thus typically prioritised over security enhancements regarding firmware and software development.

When considering to deploy SDN technology, organisations are well-advised to perform a risk-benefit analysis composed of the magnitude of the potential losses and the occurrence probability of such losses. To assess risks, threats have to be identified first. Given the large number of individual threats that may be relevant, threat groups or categories are typically used. Microsoft's STRIDE is a well-known approach to identify security design flaws and therefore viable for the security assessment of SDN. The term itself is an acronym derived from the initials of the six main threat categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS), and Elevation of Privilege. This paper deconstructs the concept of SDN into its core elements (the data plane, the control plane, and the OpenFlow protocol) and applies a STRIDE analysis to the individual components. The mechanisms of the data plane switches and communication protocols were inferred from the standard works of the ONF [1], while the control plane has been defined by analyzing several controller systems. The results of the analysis are used as first steps to build a formal SDN security architecture.

The remainder of this paper is structured as follows: Section II summarises related work in the area of SDN security analysis. Subsequently, Section III presents the results of the STRIDE application to current SDN concepts. The identified issues influence the design of a secure SDN architecture, which is described in Section IV. Section V then discusses the security properties achieved with this architecture and remaining issues which may indicate potentially fundamental security flaws in the current SDN concepts. A summary and outlook conclude the paper.

II. RELATED WORK

As SDN has gained traction, more and more security assessments and tests have been performed on the new paradigm. Kreutz, Ramos, & Verissimo, 2013 [2] identify threats which SDN may face and found several new threat vectors. In response, they propose design choices which should be considered when deploying a software-defined network but do not name or list current, suitable research solutions. This paper expands on their work and suggests practical findings. There

 ISBN: 978-1-61208-450-3

TABLE I. The threats of the STRIDE analysis summarized.

Threat	Description
Spoofing	Allows attackers to fool a system and to conceal or fake their identity. Spoofing is frequently enabled by a lack of proper authentication and verification.
Tampering	Intruders are able to compromise the integrity of transported or stored data. A malicious user could potentially alter or delete information to his advantage, without triggering an alarm or being noticed by the owner.
Repudiation	Interactors in the system are capable of denying their actions or are able to blame others. Logs and tracking systems are incomplete and can not accurately identify the perpetrator.
Information Disclosure	By abusing a vulnerability, a system might reveal sensitive data or passwords to an attacker. Eavesdropping is often correlated with preparing a sophisticated Spoofing or Tampering attack.
Denial of Service	Assets may be subject to an attack, which renders the service or system temporarily unusable to customers or users. This method has a significant financial impact and is therefore one of the most common threats.
Elevation of Privilege	This vulnerability often stems from a lack of access control. A simple user or client is able to escalate his authority in the system, which provides them with the capability to freely access restricted or classified assets.

are several attempts to specifically analyse the OpenFlow protocol. [3], [4], [5]

Using STRIDE, Klöti, Kotronis, & Smith, 2014 [6] and Brandt, Khondoker, Marx, & Bayarou, 2014 [5] utilise a similar approach as this paper. Klöti, Kotronis & Smith evaluate SDN based on Tampering, Information Disclosure, and Denial of Service, develop attack trees and perform security tests, but they do not extend the scope beyond the OpenFlow v1.0 protocol and do not model all of the six threats in depth. Brandt, Khondoker, Marx, & Bayarou evaluate various SDN protocol implementations using STRIDE and identify several security vulnerabilities. However, they specifically focus on the lack of TLS support in the 1.4 version of the OpenFlow and do not address further vulnerabilities. A recent survey of Scott-Hayward, Natarajan & Sezer, 2015 [7] addresses challenges and opportunities regarding the security of SDN and provides a comprehensive overview over security enhancements and problems in SDN. Lastly, the security research team of the ONF has reviewed the protocol and proposed amendments, which are likely to be adopted in future standards. [4] These amendments are limited to the protocol, but are taken into consideration in the final security analysis.

Building upon the previous work, this paper provides a condensed overview over current research and systematically decomposes the security of SDN into six aspects. In addition to the security assessment, requirements for developers as well as mandatory implementation guidelines are specified. These solutions and amendments are intended to propose a draft of a secure SDN architecture for network operators, which can be further evaluated.

III. STRIDE Analysis Applied to Software-Defined Networking

SDN shifts the control of the entire network to a single autonomous software system. One outcome is new flexibility in the network, but also a high level of dependency. Consequently, the architecture may harbour frequently unconsidered risks. The increased focus on software, programmability and open interfaces may introduce several new access points to an intruder. Furthermore, the central controller is a prime target for DoS and manipulation attacks, as the flow of the entire network depends on a single unit. Since the impact of a compromised unit is significantly magnified, development of SDN devices needs to be subject to continuous threat examination.

The STRIDE threat model (see Table I) is utilised to provide an overview of deficiencies and possible negligences of the concept. To construct a framework of current SDN which can be analysed, the paper drew from standard literature [8], [1] and default network configurations. This work is an abridged version and summary of the results of the thesis "Evaluating the State of Security in Software-Defined Networks". The full document can be accessed in [9].

A. Spoofing

The results of the analysis highlight that, although an attacker must use conventional methods to establish himself in the network, his succeeding capabilities are considerably extended. SDN introduces two largely novel components in the network, the controller and applications. The new logical elements are capable of exerting a great amount of power over the entire network, while also being imitable, therefore becoming a prime target. The programmability and programming interfaces potentially harbour a multitude of new security holes. Moreover, virtualisation of physical network devices, such as switches and the controller, lowers the barrier for imitation.

Traditional authentication protocols exist as a countermeasure. However, past negligence and security threat reports demonstrate that they might not be sufficient to protect controllers and switches from abuse [3], [10], [7]. The increased impact makes Spoofing a sizeable threat in SDN, more so than in legacy networks. Even assuming all trust boundaries of the data flow are secured in the network, Spoofing attempts are nevertheless feasible. In this analysis, Spoofing is thus considered a base vulnerability which enables further STRIDE threats. Security-conscious network operators are required to address this threat with special care and caution and have to deploy mechanism to automatically detect spoofed connections and devices based on suspicious behaviour.

B. Tampering

Tampering displays a similar threat pattern as Spoofing. The average access risk is not exacerbated, if authentication measures are properly implemented and the network is physically secure. However, the application and control plane reveal several new entry points for an attack. The modification of central information has a significantly larger impact on the network. Routing intelligence is not distributed and switches are dependent on a single entity maintaining the view of the network. If this database is affected, the whole network is compromised. A controller has to correctly identify corrupt and conflicting information in the same fashion as it has to notice and detect Spoofing attempts. The responsibility of security and consistency shifts to the control platform, which has to verify switch topology and application reports [11], [12].

 ISBN: 978-1-61208-450-3

TABLE II. TABULAR REPRESENTATION OF STRIDE-SPECIFIC THREATS AND SOLUTIONS IN SDN.

SDN Threat	Problems	Solutions
Spoofing	Illegitimate authentication as controller, switch or application due to negligent security measures and software faults.	Enforce mandatory and modern authentication procedures in the standard works. Ensure trustworthiness of remote or local application commands.
Tampering	Attacker may be able to overwrite controller policy and poison the central, virtual network view. The interception and altering of OpenFlow control messages has an extensive impact on the network configuration.	Implement access-control and integrity-verification mechanisms in the north- as well as the southbound interface of SDN. Significant actions are decided based on the votes of multiple, independent control elements.
Repudiation	Lack of inherent and automatic monitoring capabilities of switches and control software may enable covert operations.	SDN devices are uniquely identifiable. Logging and tracking mechanisms are automatic and secured.
Information Disclosure	The centralised information storage and query possibilities simplify network reconnaissance. Additionally, a compromised underlying server software may expose credentials and the network database.	Relocate SDN communication to separate and secured channels, the controller and the network data storage are removed from the data net.
Denial of Service	Switch functionality is dependent on a single, central controller and control channel which is susceptible to multitude of attack possibilities, such as flooding, exploits and software bugs. The routing tables of switches are limited and quickly exhaustible.	Deploy controller paired with intrusion-detection mechanisms and utilise fall-back mechanisms and element redundancy. Adopt maintenance and development procedures of conventional operating systems.
Elevation of Privilege	Network controllers accessible to multiple users may be compromised or expose information about neighbouring networks. As there is no distinction in the priority of the application commands, malicious client applications may assume full authority of a shared controller.	Shared resources must be subject to rigorous role-based access-control and separation mechanisms, while clients receive minimal trust in their operations. Software must be subject to regular audits during development.

C. Repudiation

The Repudiation threat in SDN does not differ significantly from legacy networks, as the major cryptographic protocols are theoretically supported. [1] In fact, the centralised overview amplifies the potential to trace covert communication activities and rogue devices [13], [14]. In this STRIDE analysis, repudiation issues in OpenFlow largely stem from Tampering or general implementation negligence, since authentication measures are seen as optional [3]. Nevertheless, introducing unique IDs, while informing the remaining network members of the actions of peer devices are compulsory considerations. It is recommended to meticulously document and monitor the activities of malfunctioning controllers and applications to provide a minimal capability to correctly track down or find suspicious behaviour.

D. Information Disclosure

New network components introduce new possibilities to collect data. The agile and programmable nature of an OpenFlow network facilitates Information Disclosure, as single devices can be quickly reconfigured to direct traffic over detour sniffing paths. The variance in response time aids attackers in mapping parts of the network without having to access any device. In SDN, multiple elements maintain information about the entire network in flow tables and virtualisation databases. This information may be revealed using remote queries or by gaining access to a server. While user data may be protected with TLS, it is an evident conclusion that basic SDN does not provide adequate methods to hide information about the overall network structure.

E. Denial of Service

SDN magnifies the risk of Denial of Service in the network to a great extent. Network elements drop independence for the sake of agility and ease of configuration. However, if the central intelligence fails, the entire network breaks down. The programmability and software-centric approach introduces new attack vectors and error potential, leading to failures or outages. Manipulation of the network map may result in intentional misconfiguration and traffic black holes. Furthermore, the multitude of possibilities to damage the system broadens the attack surface. Nevertheless, SDN may provide several opportunities to dynamically mitigate DoS attacks. Applications can isolate infected hosts, if they are identified in due time. Traffic can quickly be rerouted to avoid congestions. The switch meter band of OpenFlow switches is capable of automatically limiting incoming data rate, resulting in dynamic and agile protection of sensitive network areas. [1] The constant and centralised network monitoring of the controller may quickly identify anomalous behaviour. These possibilities, however, are based on the assumption that the controller is operational or utilises the necessary protective tools. OpenFlow does not include these capabilities by default. Intelligent applications and multiple or distributed controllers have to be deployed in order to guarantee reliable attack defence and scalability.

F. Elevation of Privilege

Presently, the maturity of SDN poses a problem in identifying potential risks in shared service networks. Dominant enterprise applications or deployments have not yet emerged to evaluate concrete design decisions. While Google did install a large-scale SDN data centre [15], they avoided the problem of conflicting applications using single application blocks and internal conflict resolution [16]. Additionally, no actual mechanisms to share controller resources between different network users or entities are available yet. In this context, the slicing software FlowVisor [17] is a popular solution to divide the network into various security or control domains. However, the reports on design flaws of the concept, which enable an attacker to break out of his limited view, are already present [18], [19]. Overall, it needs to be advocated that authorisation and proper permission distribution is a crucial cornerstone in the deployment of large scale software-defined network.

IV. A SECURE SDN ARCHITECTURE

The findings of the STRIDE methodology demonstrate that a SDN network combined with conventional protection can not be considered secure. Traditional security measures such as encryption, firewalls, or intrusion detection systems (IDS) have to be adapted to the new design. To guarantee a carrier-grade level of reliability new methods are already being developed and tested. SDN may provide the opportunity of sophisticated, automated defence, if the minimum requirements are fulfilled. This paper thus leverages STRIDE to sketch a feasible security

 ISBN: 978-1-61208-450-3

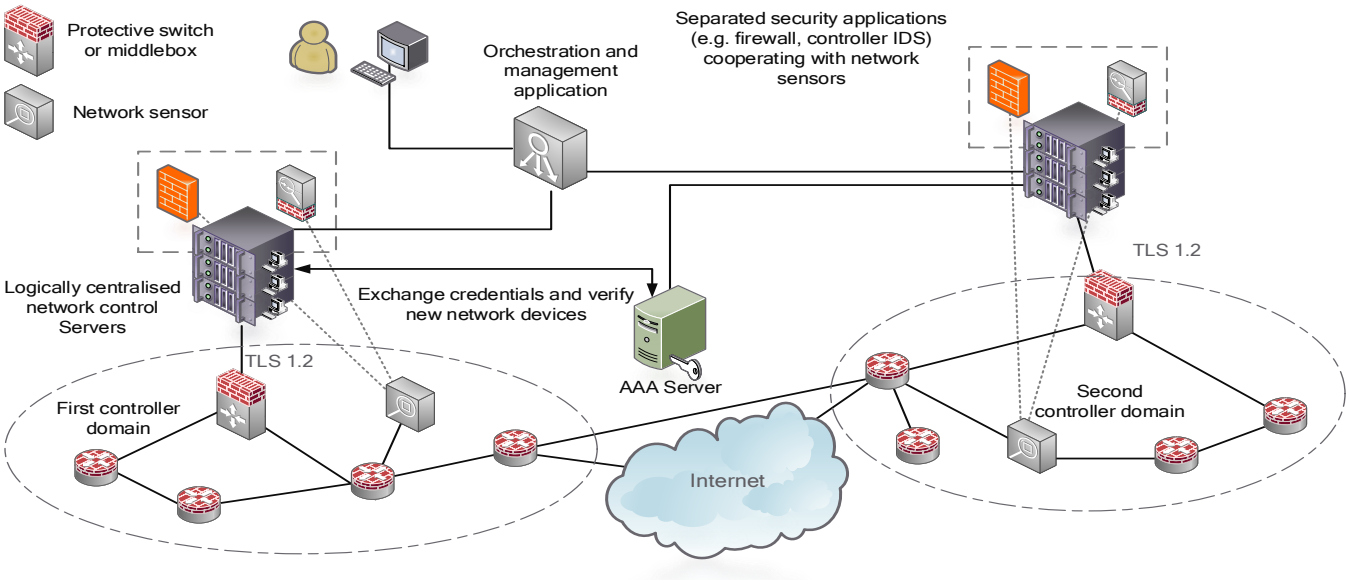


Figure 1. Sample sketch of a secure SDN design for two network domains. Multiple controllers provide a basic degree of redundancy and are kept in synchronization via a SDN application. The controllers are protected from unauthorized access using a local IDS as well as firewalls and access control.

architecture which integrates traditional and new protection solutions. The design may be utilised to assess the security potential of future SDN as well as to formulate minimal necessary security requirements for larger software-defined networks.

In order to mitigate the encountered vulnerabilities and flaws, relevant literature was consulted, advanced security solutions were inspected, and requirements and design choices which introduce sophisticated defence capabilities and network robustness were specified. Furthermore, an assessment of the ONF, which defines necessary security improvements for the OpenFlow protocol, is taken into account. [4] Suggestions include to mandate the use of security protocols, introduction of unique identifiability and a clear definition of trust and security boundaries. Table II summarises the problems and solutions identified in this survey. As a result, this work proposes a design sketch of a secure network utilising the principles suggested by Kreutz, Ramos, & Verissimo, 2014 [2] as well as feasible current security proposals. Note that performance and latency are not considered in this design, as the aim is to construct a SDN network utilising maximum possible security.

The first and absolute prerequisite in the secure system is the use authenticity and integrity for any device communication in the network, as these properties are neglected in current standard works and deployments. Any and all communication between applications, controllers and switches is mutually authenticated, while sensitive messages such as topology reports and modification messages are checked for integrity. The database of the controller itself is signed to guarantee the use of intact server data. Optionally, the control channel may be deployed out-of-band either physically or virtually in VLAN configurations enacted by the controller. It is advised to require authenticity in any SDN installation to ensure a minimal amount of security and protection of the control flow.

To avoid dependency on a single device, at least two independent controllers should be deployed in the network. They may coordinate or take over neighbouring networks in case of a malfunction. For added security, and to overrule malicious or defect controllers, every switch may connect to multiple logically centralised controllers. In this particular design, any independent domain should employ a minimum of three replicated controllers. They communicate directly or indirectly over a distributed network database, to minimise the threat of a compromised device. As diverse implementations appear to be horizontally incompatible so far, all controllers have to conform to the same type. If different controller types are to be implemented in the network, proxy layers might support the distribution and interoperability of the devices, while also reducing the load on single controllers in the network.

The control plane resides in a protected area, similar to a vital database in a conventional network. Only authenticated hosts, which are part of a physically and logically secured domain, are able to access and configure the servers. Any traffic which is not a OpenFlow communication message is filtered using the integrated flow table firewalls. Additionally, specialised DoS guard switches, e.g., Avant-Guard [20], may shield the control centre from attacks. Albeit potentially costly, it is recommended to apply out-of-band access of management or to use security applications.

Remote applications and hosts trying to access the server zone are verified based on location and identity, using AAA servers and control algorithms. They are also limited in rights, network view and action scope. Security and latency-intensive applications may be packaged directly on the control server but are strictly executed in a separate process and memory space. Higher-privileged applications are able to override lower-tier decisions, with the administrator applications possessing complete configuration rights.

Administrator applications report the state and log of all controllers and switches in the network and track actions of

 ISBN: 978-1-61208-450-3

the single devices and applications. With those middleboxes found to be yet irreplaceable, the control servers may be protected over intrusion detection or stateful firewall systems. Nevertheless, to quickly identify and resolve attacks in the entire network, switches could mirror traffic to selected inspection servers. The detection systems report the results to the controller, which swiftly reconfigures the network to isolate the affected sections. Additionally, the controller might be capable of identifying suspicious network behaviour based on packet patterns. Any events and anomalies in the network are reported to the management application or a dedicated Security Information and Event Management (SIEM) system.

In general, it is advised to block off the network from networks of a lower security tier. It should be divided into varying protective zones by distributing a fine-grained firewall in the network and authenticating any new host in the network. However, firewalls might still be required, as the flow table space of switches is limited and stateful firewalls are only functional via the control intelligence. To guarantee longevity and the latest secure firmware, controller could be exchanged using a live-swapping mechanism such as HotSwap [21] or live-patched by replacing single modules.

V. Discussion

After surveying the current security potential and possible opportunities of the SDN architecture, a second security analysis is conducted to evaluate the security potential of SDN as a future technology. It serves as an overview of the theoretical state of security of SDN based on current research. Figure 1 provides a graphical representation of the various methods which might be possible to secure a SDN network.

A. Spoofing

Although new elements which may be spoofed are introduced, Spoofing is sufficiently preventable via authentication and access control mechanisms. Considering the design postulations of the ONF Security Project [4], which is likely to influence future standards, it can be assumed that a deployment-ready software-defined network will utilise TLS or similar authentication. Furthermore, SDN provides an opportunity in recognising and removing spoofed devices as the controller can autonomously identify irregularities in the network. The lack of application authenticity is a threat which has not been considered by many developers, but functional countermeasures are already actively being developed [16], [22].

While attacks to spoof the host topology and redirect traffic are already present and published [23], [24], they are preventable with the use of message integrity and various security applications installed in the control plane.

B. Tampering

Since the virtual network view and databases can be modified, Tampering of data is a greater risk. It is a significant danger in environments such as SDN, which rely on a virtual view and central database. A slight change substantially affects the network and this threat must be addressed accordingly. Authentication and data integrity algorithms as demanded by the ONF [4] may protect the control traffic and data sufficiently. Additionally, democratic approaches in the control plane may override the decisions of a mislead or malicious device. It is crucial to avoid dependence on a single control station in SDN, if security and network availability are valued. Lastly, harmful influence from applications and clients may be restricted using authorisation and role-based access control in the northbound interface, while simultaneously isolating the controller from the underlying system as well as the applications.

C. Repudiation

This threat is not exacerbated in software-defined networks. However, many new deniable actions of the autonomous applications and controllers emerge. An automatic and instant logging mechanism, unique identification of individual behaviour, and monitoring of control processes reduce the possibility to hide or deny malicious actions. Furthermore, the overview of the entire network state, traffic flow, and access control establishes a comprehensive tool set for attack forensics. These surveillance possibilities are an advantage of the new architecture.

D. Information Disclosure

Similar to Tampering, the risk of Information Disclosure gains new significance in SDN. The network relies on a central information base which can be accessed over various interfaces and queries. This store contains ample data about topology, QoS policy, and restricted areas. Extracting this information gives an attacker substantial knowledge about assets, security appliances, and the location of sensitive data. However, if access to the control plane and channel is safely restricted, the sensitive information is moved out of reach. Relocating the controller into a secure and restricted zone is a core design choice. Access has to be limited to physical and authorised applications or administrative stations and the control channel should be secured using dedicated VLAN as well as out-of-band communication.

These implementations are achievable and prevent the controlling software from unintentionally leaking information. Switches of the data plane may expose their flow tables over side-channel attacks [6], but the large amount of traffic needed to acquire this information is detectable by an integrated controller IDS, e.g., tools such as the SPHINX [25] proposal. If the controller is secure and detached from the data network, it is also capable of reducing the transparency of the network by using dynamic proxy approaches such as OpenFlow Random Host Mutation (OF-RHM) [26].

In summary, Information Disclosure is preventable in the secure SDN design, if the control channel and plane are appropriately guarded and entirely separated from the intranet and generic data traffic.

E. Denial of Service

Denial of Service is major problem of SDN, as an attack on the control plane paralyses the entire network. The presented countermeasures of the secure design, i.e., restricting the access to the controller, implementing a dedicated IDS, and building a protection ring, shield the device but may be circumvented. Due to the focus on software and the control bottleneck, a multitude of possibilities arises to incapacitate a central switch or controller, ranging from simple flooding to the use of poison packets or malicious applications. The key to protection is isolation, replication of essential assets, and synchronisation, all of which assure fall-back guarantee and a sufficient degree of availability and reliability. Use of distributed data stores is problematic, as these may be susceptible

 ISBN: 978-1-61208-450-3

to service failure or abuse. A currently prevalent problem is the potential limitation of the amount of flow table entries in OpenFlow hardware switches [27]. SDN requires many specific entries on a single switch for fine-grained network management. Therefore, switches may operate on the brink of table exhaustion in large networks and thus may become an easy target for attackers. Denial of Service as a threat is amplified in SDN and is still a ubiquitous risk in the secure design. Although it is possible to prevent most of the dangers with the discussed methods, these new measures might again introduce new vulnerabilities. The threat requires sophisticated protective measures and careful consideration in order to fully protect the sensitive controllers and the simple switches in the network and to guarantee high availability.

F. Elevation of Privilege

The last aspect is an entirely new factor, which has to be observed and studied when deploying a software-defined network as a service. Due to the utilisation of operating system principles in SDN, it is possible for unauthorised clients to access the virtualised and shared network resources and enact configurations which they are not entitled to. Role- or permission-based access control, verification, and separation mechanisms, such as FlowVisor, address these concerns. Nevertheless, breaking out of the virtualised, restricted box and traversing prohibited domains is a constant hazard when providing network services. Clients must not be trusted and have to be restricted and strictly monitored when accessing the public control plane. Research on SDN and the Network as a Service (NaaS) concept is still in its infancy and solutions may risk neglect or miss security flaws during this development process. In order to prevent security leaks, the same rigour and meticulous process as in the development of current operation systems has to be applied to the controller and virtualisation deployment and real-world operations.

VI. Summary and Outlook

Software-Defined Networking is a pivotal new paradigm for data centre networks and on the verge to reach out to the Internet backbone infrastructure. Security thus understandably becomes an important aspect, which is currently addressed by both research and industry, e. g., as part of the Open Networking Foundation's recent security principles and practices document. [4]

However, a closer security analysis of the current state of the art in SDN, as the STRIDE-based one presented in this paper, quickly reveals a wide range of SDN-specific threats, which have not yet been counteracted adequately. Some of them are inherently tied to SDN design principles, such as controllers becoming potential central attack targets; others are inherited from the underlying infrastructure, e. g., the susceptibility to Spoofing; practical threats are also related to the implementation maturity, such as the potential lack of isolation between applications running on the same SDN controller. Based on the results of this analysis, this paper suggested key factors and constraints of a secure SDN architecture. It emphasises the role of authenticity and integrity controls for the involved components and the management protocol messages exchanged between them. A key element of the architectural design is to ensure that security measures not only prevent, but also detect attempted and successful attacks on the SDN components. Objective improvements in comparison to traditional networking components' firmware implementation will require the adoption of modern methods, such as live-patching or live-swapping. It is also worth noting that securing the management communication still has to rely on well-established traditional concepts, such as out-of-band management or at least separate management VLANs. Furthermore, solutions to prevent flow table flooding, e. g., as a result of DoS attacks, will need to be designed and deployed.

In our future work, we will address the use of SDN in distributed data centres, which are set up closer to customers and end users to store and process huge amounts of data where it is generated and required. Based on methods, such as location awareness and automated local routing mechanisms, service level agreements and end-to-end service quality guarantees will become possible and serve as the basis for the secure and high-performance operations of virtualised network functions and networks-as-a-service.

Acknowledgment

Parts of this work has been funded by the German Ministry of Education and Research (FKZ: 16BP12309). The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful comments on previous versions of this paper. The MNM-Team, directed by Prof. Dr. Dieter Kranzlmüller and Prof. Dr. Heinz-Gerd Hegering, is a group of researchers at Ludwig-Maximilians-Universität München, Technische Universität München, the Universität der Bundeswehr München, and the Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities. See www.mnm-team.org.

References

[1] Technical Specification: OpenFlow Specification 1.5.0, 6th ed., Open Networking Foundation, 2015.

[2] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards Secure and Dependable Software-defined Networks," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN'13, 2013, pp. 55–60.

[3] K. Benton, L. J. Camp, and C. Small, "OpenFlow Vulnerability Assessment," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN'13, 2013, pp. 151–152.

[4] Project Security, "Principles & Practices for Securing Software-Defined Networks applied to OFv1.3.4 Ver 1.0," Open Networking Foundation, Tech. Rep., 2014.

[5] M. Brandt, R. Khondoker, R. Marx, and K. Bayarou, "Security analysis of software defined networking protocols - OpenFlow, OF-Config and OVSDB," in Proceedings of the Fifth IEEE International Conference on Communications and Electronics, ser. ICCE'14, 2014, pp. 23–30.

[6] R. Klöti, "OpenFlow: A Security Analysis," Master's thesis, Eidgenössische Technische Hochschule Zürich, 2013.

[7] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," IEEE Communications Surveys and Tutorials, 2015, p. 1.

[8] Technical Recommendation: SDN Architecture, 1st ed., Open Networking Foundation, 2014.

[9] F. Ruffy, "Evaluating the State of Security in Software-Defined Networks," Master's thesis, Ludwig-Maximilians-Universität München, 2015.

[10] D. an Romão, N. van Dijkhuizen, S. Konstantaras, and G. Thessalonikefs, "SSN Project Report: Practical Security Analysis of Openflow," SSN Project Report, 2013.

 ISBN: 978-1-61208-450-3

[11] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A Robust, Secure, and High-performance Network Operating System," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS'14, 2014, pp. 78–89.

[12] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a Secure Controller Platform for Openflow Applications," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN'13, 2013, pp. 171–172.

[13] J. François and O. Festor, "Anomaly Traceback using Software Defined Networking," in International Workshop on Information Forensics and Security, ser. WIFS'14, 2014.

[14] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN be your eyes: Secure forensics in data center networks," in Proceedings of the 2014 NDSS Workshop on Security of Emerging Network Technologies, ser. SENT14, 2014.

[15] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-deployed Software Defined Wan," in Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM, ser. SIGCOMM'13, 2013, pp. 3–14.

[16] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the Software-Defined Network Control Layer," in Proceedings of the 2015 Network and Distributed System Security Symposium, ser. NDSS'15, 2015.

[17] R. Sherwood, G. Gibb, K. Yap, M. Casado, N. Mckeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," OpenFlow Switch Consortium, Tech. Rep., 2009.

[18] W. You, K. Qian, X. He, Y. Qian, and L. Tao, "Towards Security in Virtualization of SDN," in Proceedings of the International Conference on Computer Communications and Networks Security, ser. ICCCNS'14, 2014, pp. 1419–1422.

[19] V. Costa and L. M. K. Costa, "Vulnerabilities and solutions for isolation in flowvisor-based virtual network environments," Journal of Internet Services and Applications, vol. 6, no. 1, 2015.

[20] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow Management in Software-defined Networks," in Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS'13, 2013, pp. 413–424.

[21] L. Vanbever, J. Reich, T. Benson, N. Foster, and J. Rexford, "HotSwap: Correct and Efficient Controller Upgrades for Software-defined Networks," in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN'13, 2013, pp. 133–138.

[22] S. Scott-Hayward, C. Kane, and S. Sezer, "OperationCheckpoint: SDN Application Control," in Proceedings of the 22nd International Conference on Network Protocols, ser. ICNP 22, 2014, pp. 618–623.

[23] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures," in Proceedings of 2015 Annual Network and Distributed System Security Symposium, ser. NDSS'15, 2015.

[24] M. Antikainen, T. Aura, and M. Saerelae, "Spook in Your Network: Attacking an SDN with a Compromised OpenFlow Switch," in Secure IT Systems, ser. Lecture Notes in Computer Science, 2014, vol. 8788, pp. 229–244.

[25] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in Proceedings of 2015 Annual Network and Distributed System Security Symposium, ser. NDSS'15, 2015.

[26] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking," in Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ser. HotSDN'12, 2012, pp. 127–132.

[27] M. Kuzniar, P. Peresini, and D. Kostic, "What You Need to Know About SDN Flow Tables," Lecture Notes in Computer Science (LNCS), 2015, pp. 347–359.

 ISBN: 978-1-61208-450-3

Multi-criteria based Optimization of Placement for Software Defined Networking Controllers and Forwarding Nodes

Eugen Borcoci, Tudor Ambarus, Marius Vochin
University POLITEHNICA of Bucharest - UPB
Bucharest, Romania
Emails: eugen.borcoci@elcom.pub.ro, tudorambarus@yahoo.com, mvochin@elcom.pub.ro

***Abstract* — Placement of Software Defined Networking (SDN) controllers and forwarders in large networks contexts is still a research open issue, given the different network contexts, providers' policies and possible optimization criteria. Multi-criteria decision algorithms can provide valuable solutions. This paper is an extension of a previous preliminary work, considering here large network environments and the additional problem of forwarding nodes assignment to SDN controllers.**

Keywords — Software Defined Networking; Multi-criteria optimizations; Controller placement; Forwarding nodes assignment; Reliability;

I. INTRODUCTION

Emergent Software Defined Networking (SDN) architectures and related technologies are of high interest for industry and operators, in wire-line and wireless networks and cloud computing environments, [2][3]. This paper considers the case of a Wide Area Network (WAN) owned by an operator and/or a Network/Service Provider (NP/SP).

For large SDN-controlled networks, multi-controller solutions are proposed to solve the scalability problems, related to SDN control centralization principle [5][6][7]. Flat or hierarchical organizations for multi-controller SDN are suggested in [6][7] (in the subsequent text, by "controller" it is understood a geographically distinct controller location). The data forwarding network nodes (called also "forwarders" or simply, "nodes") should be allocated to some controllers in a proactive or reactive way. Some design problems are: *What is the optimal number and placement of the controllers? How to allocate the forwarder nodes to controllers?*

The controller placement problem is a NP-hard one [9]. So, different solutions have been proposed, with specific optimization criteria, targeting performance in failure-free or more realistic scenarios. However, some criteria could lead to different solutions; so, a multi-criteria global optimization could be attractive. Some specific criteria could be defined as to: (a) maximize the *controller-forwarder* or *inter-controller communication* throughput, and/or reduce the latency of the path connecting them; (b) limit the controller overload (load imbalance) by avoiding too many forwarders per controller; (c) find an optimum controllers' placement and forwarder-to-controller allocation, offering a fast recovery after failures (controllers, links, nodes).

Also, other specific optimization goals could be added to the above list, depending on specific context (wire-line, wireless/cellular, cloud computing and data center networks) and on some specific business targets of the Service Provider.

The paper [1] provides a contribution on *multi-criteria optimization algorithms* for the controller placement problem. The target was *not to develop specific algorithms* to find an optimum solution for *a single given criterion* (several other studies already did that) but to *achieve an overall optimization on controller placement*, by applying *multi-criteria decision algorithms (MCDA)* [10]. The input of MCDA is the set of candidates (an instance of controller placement was called a *candidate solution*). Simple examples have been analyzed, proving the usefulness of the approach.

This paper is an extension of [1] by constructing a software simulation model; it considers larger realistic topologies, variation of the MCDA criteria and optimized allocation of the forwarders to controllers. Simulation experiments and novel results are presented.

The paper is organized as follows. Section II is an overview of related work. Section III revisits several metrics and algorithms used in optimizations and presents some of their limitations. Section IV develops the framework for MCDA-RL (reference level variant) to select the best controller placement solution. Section V presents a set of simulation experiments performed and the results obtained. Section VI presents conclusions and future work.

II. SDN CONTROLLER PLACEMENT -RELATED WORK

This short section is included for self containment of this paper. A more comprehensive overview on some previously published work on controller placement in SDN-managed WANs is given in [1]. The basic problem to be solved is on the number of controllers and their placement in a given network. The goal is to provide enough performance (e.g., low delay for controller-forwarder communications) and also create robustness to controllers and/or network failures.

The works [8][9] have shown that the above problem is theoretically not new. If *latency* is taken as a metric, the

 ISBN: 978-1-61208-450-3

problem is similar to a known one, namely, the *facility or warehouse location problem*, solved, e.g., by using Mixed Integer Linear Program (MILP) tools.

Heller et al. [9] have shown that it is possible to find optimal solutions for realistic network instances, in failure-free scenarios, by analyzing the entire solution space, with off-line computations (the metric is latency). Going further, the works [8][11][14][15][16] additionally considered the resilience as being important with respect to events like: *controller* failures, *network links/paths/nodes* failures, *controller overload* (load imbalance). The *Inter-Controller Latency* is also important and generally it cannot be minimized while simultaneously minimizing controller-forwarders latency; a tradeoff solution could be the answer.

The works [8][15] developed several placement algorithms for some real topologies, trying to improve the reliability of SDN control, but still keep acceptable latencies. The controller instances are chosen as to minimize connectivity losses; connections are defined according to the shortest path between controllers and forwarding devices. Muller et.al. [16] try to eliminate some restrictions of previous studies, like: single paths, processing (in controllers) of the forwarders requests only *on-demand* and some constraints imposed on failover mechanisms.

As stated previously, *this paper does not aim to develop a new algorithm for optimized controller placement, based on a given particular metric, but extends a previous overall optimization work, while using multiple criteria.* The general part of the problem, exposed in [1] is summarized here for sake of self-containment.

III. SUMMARY OF CONTROLLER PLACEMENT METRICS ALGORITHMS

This section is a short presentation of a few typical metrics and optimization algorithms for controller placement. A more extended presentation can be found in [1]. Considering a particular metric (criterion) an optimization algorithm can be run, as in [8][9][11][16]. This paper goal is not to develop a new particular algorithm - but to search for a global optimization.

A. Performance-only related metrics (failure-free scenarios)

The network is represented by an undirected graph G(V, E), where V,E are the sets of nodes and edges, respectively and n=|V| is the number of nodes. The edges weights represent an additive metric (e.g., *propagation latency* [9]). The controllers will be co-located to some network nodes.

A simple metric is d(v, c): *shortest path* distance from a forwarder node $v \in V$ to a controller $c \in V$. In [9], two kinds of latencies are defined, for a particular placement C_i of controllers, where $C_i \subseteq V$ and $|C_i| \leq |V|$. The number of controllers is limited to $|C_i| = k$ for any particular placement C_i. The set of all possible placements is denoted by $C = \{C_1, C_2 \ldots\}$. One can define, for a given placement C_i:

Worst_case_latency:

$$L_{wc} = \max_{v \in V} \min_{c \in C_i} d(v, c) \quad (1)$$

Average_latency:

$$L_{avg}(C_i) = \frac{1}{n} \sum_{v \in V} \min_{c \in Ci} d(v, c) \quad (2)$$

The algorithm should find a placement C_{opt}, where *either average latency* or *the worst case latency is minimized.*

The limitations of this optimization process consist in: static values assumed for latencies, despite that delay is a dynamic value in IP networks; only free-failure case are considered; no upper limit on the number of forwarders assigned to a controller; not taking into account the inter-controller connectivity. Another possible metric to be considered in failure-free case is *Maximum cover,* [9][17]. The algorithm should find a controller placement, as to *maximize the number of nodes within a latency bound*, i.e., to find a placement of *k* controllers such that they cover a maximum number of forwarder nodes, while each forwarder must have a limited latency bound to its controller.

B. Reliability aware metrics

More realistic scenarios consider controller and/or network failures events. The optimization process aims now to find trade-offs to preserve a convenient behavior of the overall system in failure cases.

(1) Controller failures (cf): the work [11] observes that the node-to-controller mapping can change in case of controller failures. So, the latency-based metric should consider both the distance to the (primary) controller and the distance to other (backup) controllers. For a placement of a total number of *k* controllers, the failures are modeled by constructing a set C of scenarios, including all possible combinations of faulty controller number, from 0 of up to k - 1. The resulting maximum latency will be:

Worst_case_latency_cf:

$$L_{wc-cf} = \max_{v \in V} \max_{C_i \in C} \min_{c \in C_i} d(v, c) \quad (3)$$

The optimization algorithm should find a placement which *minimizes the expression* (3).

Note that in failure-free case, the optimization algorithm tends to rather equally spread the controllers in the network, among the forwarders. To minimize (3) and considering worst case failure, the controllers tend to be placed in the center of the network. Thus, in a worst case a single controller can take over all control. However, the scenario supposed by the expression (3) is very pessimistic; a large network could be split in some regions/areas, each served by a primary controller; then some lists of possible backup controllers can be constructed for each area, as in [16].

The conclusion is that an optimization trade-off should be found, for the failure-free or failure cases. This, once again, shows that a multi-criteria approach is attractive.

(2) Nodes/links failures (Nlf):

 ISBN: 978-1-61208-450-3

Links or nodes failures can cause some forwarders to lose access to all controller. An objective could be to find a controller placement that minimizes the number of nodes possible to enter into controller-less situations, in various scenarios of link/node failures. A realistic assumption is to limit the number of simultaneous failures at only a few (e.g., two [11]). If more than two arbitrary link/node failures happen simultaneously, then the topology can be totally disconnected and optimization of controller placement would be no longer useful.

For *any given placement* C_i of the controllers, an additive integer value metric $Nlf(C_i)$ could be defined, as below: consider a failure scenario denoted by f_k, with $f_k \in F$, where F is the set of all network failure scenarios (suppose that in an instance scenario, at most two link/nodes are down); initialize $Nlf_k(C_i) = 0$; then for each node $v \in V$, add one to $Nlf_k(C_i)$ if the node v has no path to any controller $c \in C_i$ and add zero otherwise; compute the maximum value (i.e., consider the worst failure scenario). One obtains the formula (4) where *k* covers all scenarios of F.

$$Nlf\ (C_i) = \max\ \ Nlf_k(C_i) \tag{4}$$

The *optimization algorithm* should find a *placement which minimizes (4).* It is expected that increasing the number of controllers, will decrease the *Nlf* value. However, the optimum solution based on the metric (4) could be very different from those provided by the algorithms using the metrics (1) or (2).

(3) Load balancing for controllers

A good balance of the node-to-controller distribution is desired. A metric $Ib(C_i)$ will measure the degree of imbalance of a given placement C_i as the *difference between the maximum and minimum number of forwarders nodes assigned to a controller*. If the failure scenarios set S is considered, then the worst case should evaluate the maximum imbalance as:

$$Ib(C_i) = \max_{s \in S} \{ \max_{c \in C_i} n_c^s - \min_{c \in C_i} n_c^s \} \tag{5}$$

where n_c^s is the number of forwarder nodes assigned to a controller c. Equation (5) takes into account that in case of failures, the forwarders can be reassigned to other controllers and therefore, the load of those controllers will increase. An *optimization algorithm* should find that *placement which minimizes the expression (5).*

(4) Multiple-path connectivity metrics

One can exploit the possible multiple paths between a forwarder node and a controller [16], hoping to reduce the frequency of controller-less events, in cases of failures of nodes/links. The goal in this case is to maximize connectivity between forwarding nodes and controller instances. The metric is defined as:

$$M(C_i) = \frac{1}{|V|} \sum_{c \in C_i} \sum_{v \in V} ndp(v,c) \tag{6}$$

In (6), *ndp(v,c)* is the *number of disjoint paths* between a node v and a controller c, for an instance placement C_i. An *optimization algorithm should find the placement* C_{opt} *which maximizes* $M(C_i)$.

C. Inter-controller latency (Icl)

The inter-controller latency has impact on the response time of the inter-controller mutual updating. For a given placement C_i, the *Icl* can be given by the maximum latency between two controllers:

$$Icl(C_i) = \max\ d(c_k, c_n) \tag{7}$$

Minimizing (7) will lead to a placement with controllers close to each other. However this can increase the forwarder-controller distance (latency) given by (1) and (2). Therefore, a trade-off is necessary, *thus justifying the necessity to apply some multi-criteria optimization algorithms, e.g., like Pareto frontier - based ones* [10].

IV. MULTI-CITERIA OPTIMIZATION ALGORITHM APPLIED FOR CONTROLLER PLACEMENT PROBLEM

While particular metrics and optimization algorithms can be applied (see Section III), some criteria lead to partially contradictory controller placement solutions. As shown in introduction (and [1]) an MCDA can provide an answer. It allows selection of a trade-off solution, based on several criteria. Note that partially such an approach has been already applied by Hock et.al. [11], for some combinations of the metrics defined there (e.g., max. latency and controller load imbalance for failure-free and respectively failure use cases).

This paper uses the same variant of MCDA implementation as in [1], i.e., the *reference level (RL) decision algorithm* [10] as a general way to optimize the controller placement, while considering an arbitrary number metrics. The MCDA-RL selects the optimal solution based on normalized values of different criteria (metrics).

Given *m* objectives functions (values to be minimized) one can identify the solutions as points in an objectives space R^m, where decision parameters/variables are: v_i, i = 1, ..m, with $\forall i$, $v_i \geq 0$; the image of a candidate solution is $Sl_s=(v_{s1},v_{s2}, ..,v_{sm})$, represented as a point in R^m and where S = number of candidate solutions.

The basic MCDA-RL [10], defines two reference parameters: r_i =*reservation level*=the upper limit, which the actual decision variable v_i of a solution should not cross; a_i=*aspiration level*=the lower bound beyond which the decision variables (and therefore, the associate solutions) are seen as similar. Applying these for each decision variable v_i, one can define two values named r_i and a_i, by computing among all solutions s = 1, 2, ..S:

$$\begin{aligned} r_i &= \max\ [v_{is}],\ s = 1, 2, ..S \\ a_i &= \min\ [v_{is}],\ s = 1, 2, ..S \end{aligned} \tag{8}$$

In [10], modifications of the decision variables are proposed: *replace each variable* with *distance from it to the reservation level*: $v_i \rightarrow r_i - v_i$; (increasing v_i will decrease the distance); normalization is also introduced, in order to get non-dimensional values, which can be numerically compared. For each variable v_{si}, a ratio is computed:

$$v_{si}' = (r_i - v_{si})/(r_i - a_i), \quad \forall s, i \tag{9}$$

The factor $1/(r_i - a_i)$ - plays also the role of a weight. The variable having high dispersion of values (max – min) will have lower weights and so, greater chances to determine the minimum in the next relation (10). So, if the values *min, max* are rather close to each other, then solution is chosen is "good", w.r.t. that respective decision variable.

The basic MCDA-RL algorithm steps are:

Step 0. Compute the matrix $M\{v_{si}'\}$, $s=1\ldots S$, $i=1\ldots m$

Step 1. Compute for each candidate solution *s*, the minimum among all its normalized variables v_{si}':

$$\min_s = \min\{v_{si}'\}; i = 1...m \tag{10}$$

Step 2. Make selection among solutions by computing:

$$v_{opt} = \max\{\min_s\}, s = 1, ..S \tag{11}$$

Formula (10) selects for each candidate solution *s*, the worst case, i.e., the closest solution to the reservation level (after searching among all decision variables). Then the formula (11) selects among the solutions, the best one, i.e., that one having the highest value of the normalized parameter. One can also finally select more than one solution (quasi-optimum solutions in a given range). The network provider might want to apply different policies when deciding the controller placement; so, some decision variables could be "more important" than others. A simple modification of the algorithm can support a variety of provider policies. The new normalized decision variables will be:

$$v_{si}' = w_i(r_i - v_{si})/(r_i - a_i) \tag{12}$$

where $w_i \in (0,1]$ is a weight (priority), depending on policy considerations. Its value can significantly influence the final selection. A lower value of w_i represents actually a higher priority of that parameter in the selection process.

The controller placement computing procedure (given the graph, link costs/capacities, constraints, desired number of controllers, etc.) is composed of two phases:

(1)*Phase 1*: Identify the parameters of interest, and compute the *values of the metrics for all possible controller placements*, using specialized algorithms and metrics like those defined in formulas (1) - (7). This Phase will produce the set of candidate solutions (i.e., placement instances). This procedure could be time consuming (depending on network size) and therefore, could be performed off-line [9].

(2)*Phase 2*: MCDA-RL: define r_i and a_i, for each decision variable; eliminate those candidates having parameter values out of range defined by r_i; define – if wanted – convenient weights w_i for different decision variables; compute the normalized variables (formula (12)); run the MCDA Step 0, 1 and 2 of the (formulas (10) and (11)).

The decision variables could be among those of Section III, i.e.: *Worst_case (1)* or *Average* (2) *latency* (failure-free case); *Worst_case_latency_cf (3); Nodes/links failures (Nlf) (4); Controller Load imbalance (5);Multi-path connectivity metric (6); Inter-controller latency (7).*

For a particular problem, a set of relevant variables should be defined. For instance, in a high reliable network environment one could consider only failure free metrics.

V. USE CASE STUDIES AND SIMULATION RESULTS

A proof of concept simulation program (written in Python language [18]) has been constructed by the authors, to validate the MCDA–RL based controller assignment procedure and allocation of forwarders to controllers.

The input information (of the current program version) are: the network (overlay or physical) topology graph and link costs (it is supposed an additive metric representing the estimated delays, or 1/bandwidth on network links); the number of controllers wanted; decision parameters – e.g., some of the metrics (1) – (7); priorities/weights (policy derived) assigned to the decision variables; the set of possible solutions (e.g., possible placement of the controllers - candidate solutions- resulted from some other specific metric algorithms). Note that the topology and costs can be deterministic or randomly generated. The program works on all possible placements and then selects the best solution based on weighted MCDA-RL.

In [1], very simple topologies have been considered as examples. In this study, real networks are considered (see Figure 1), taken from [12]. Note that usually the backbone nodes are not supposed to be simple forwarders, but such topology provides a relevant network graph, to illustrate the solving of the SDN controllers placement problem.

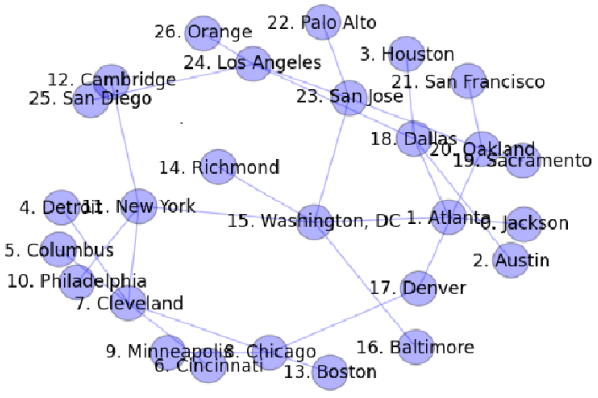


Figure 1. Real network topology example [19]

The average bandwidth of each link is estimated to 45 Mbps. From the computation point of view, one can estimate that such costs are equivalent to an additive metric normalized *to link_cost = 1*. In an equivalent way, one can assume that link latency is also normalized and can be measured relatively in few integer units. The network can be represented by an abstract graph, where each link has a cost equal to 1.

A. Controller placement

Suppose that for this network the metrics of interest and decision variables are: d1: ***W**orst latency (1,)* d2: ***A**verage latency* (2), (failure-free case); (failure-free case); d3: ***I**nter-controller latency (7).*

The reference levels are defined as (8) and in the first set of simulation we selected: r_1=6, a_1=0; r_2=3, a_2=0; r_3=6, a_3=0. For the first experiment we have chosen equal weights of the decision variables: $w_i = w = 1$, $w_2 = a = 1$, $w_3 = i = 1$.

The total number of controllers is k=2. The first set of results illustrate the best controller placement.

In Phase 1 one should compute the metrics (1), (2) and (7) and then generate the populations of candidate solutions for controller placement. To do this, it is first necessary to know the distances between different nodes while adopting the *shortest path* approach. By using the classic *Dijkstra algorithm* the shortest distances from each node to any of others (bi-directional links are supposed) can be computed (these are shortest path trees). Then for each candidate placement solution the metrics (1), (2) and (7) are computed.

Note that for large networks the Phase 1 computing time could be large, given that a complete population of solutions should be generated. However, in this study such computations are considered to be offline, i.e., the objective is not to optimize the algorithm from this point of view (see Heller [11], for discussion of such aspects).

In Phase 2, the MCDA-RL is executed (launching command is *[atudor@localhost mcda]$ python mcda.py -w 1 -a 1 -i 1]*). The results are: *Optimum Ci placement is Ci = 176 (among the total number of solutions which is* C^{2}_{26} *= 351); Controllers are placed in node 23 and node 7.*

This is the best trade-off solution, while considering the three objectives defined by the metrics (1) (2) and (7) (see Figure 2).

B. Allocation of forwarder nodes to controllers

Once the placement of controllers is known, the allocation of the forwarder nodes to controllers should be performed. The solution applied here is a constructive one, given that a single natural criterion could be applied – i.e., to select for a forwarder the closest controller. Additionally, the allocation procedure might have a constraint: a limit for the maximum numbers of forwarders allowed to be allocated to a single controller, in order to prevent imbalances.

As an example, considering the shortest path criterion and placement of two controllers in nodes (C7, C23), the allocation of the forwarders to these controllers is shown in Figure 2, marked by different colors. No limit on the number of forwarders assigned to a given controllers have been considered in this scenario. Therefore, there is some imbalance (16 forwarders assigned to C23 and only 9 to C7). If the imbalance is considered to be too high then, additional optimizations are needed.

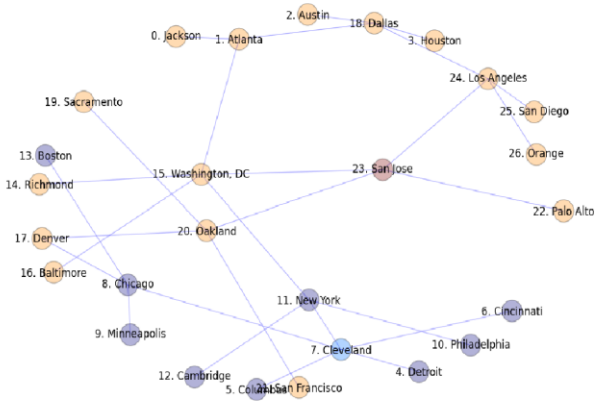


Figure 2. Controller placement and forwarder allocation (equal weights of decision variables: $w_1 = w_2 = w_3 = 1$); *k= 2 controllers*

C. Applying different weights – driven by policies

If policies should be enforced by the Network/Service Provider, then different weights can be assigned to the decision variables (see formula (12)). As an example, let us suppose that a low inter-domain latency should have higher priority than the latencies forwarders-controllers. In this case the metric (7) should have a weight <1; an example is given below, where one has the values: w= 1, a=1, i= 0.5 (the last weight is assigned to the metric (7)). The MCDA-RL will select as best, another solution for controller placement, i.e., C11 and C15 as presented in Figure 3. Note that C11 and C15 are closed to one another (C11-C15 distance = 1). However, the worst and average latencies in such case will be higher than for Figure 2 solution.

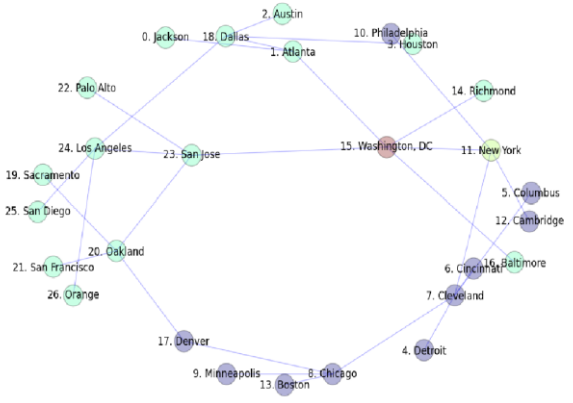


Figure 3. Controller placement and forwarder allocation with non-equal weights of decision variables: $w_1 = w_2 = 1$; $w_3 = 0.5$; *k= 2 controllers*

D. Extension of set of objective functions

In this example, the set of metrics is more rich: in additionally to the previous three metrics, one considers the load imbalance metric (formula 5), with reference levels defined as r=6, a= 0. Also, backup controllers are assigned for each primary controller.

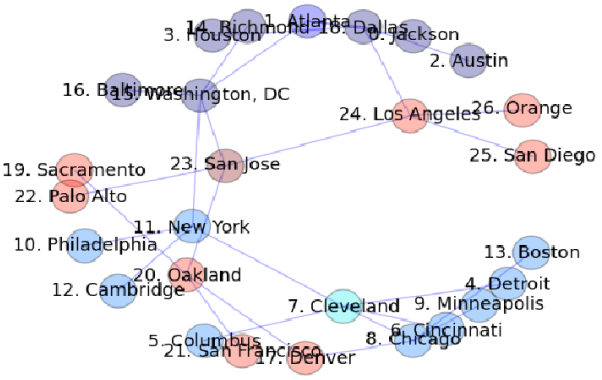


Figure 4. Controller placement and forwarder allocation with different weights of decision variables: $w_1= w_2= w_3=1; w_3=0.5; k= 3$ controllers; backup controllers are computed

To give more priority to the load imbalance metric, the weights have been selected as w_1=w=1; w_2=a=1; w_3=i=1; w_1=l=1 (for the load imbalance metric). The selected best controller placement is: C1, C7, C23. One can see (Figure 4) that allocation of forwarders to controllers is rather balanced: C1, C7, C23 have respectively 7, 9, 8 forwarders assigned to them. Also, the primary controllers C1, C7, C23 have as backup ones respectively C23, C1, C1.

VI. Conclusions and Future Work

This paper extended the study [1], on using multi-criteria decision algorithms (MCDA) to optimally select among several controller placements solutions in WAN SDN, based on weighted criteria. The MCDA-RL can produce a tradeoff (optimum) result, while considering several criteria, part of them even being partially contradictory. The method proposed here is general and can be applied in various scenarios (including failure-free assumption ones or reliability - aware), given that it achieves an overall optimization. In this study, a simulation program has been constructed and real network topologies considered. The optimum controller placement has been found, while different weights policy-driven have been introduced. Also, forwarder-controller mapping optimization and backup controller selection have been also considered. The examples given demonstrate the flexibility of the approach in selecting the best solution while considering various criteria.

Future work will be done to apply the method proposed to other – metrics, considering multi-path approach for forwarder-controller paths hierarchical networks and studying the static/dynamic aspects of this approach.

References

[1] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On Multi-controller Placement Optimization in Software Defined Networking -based WANs", The International Symposium on Advances in Software Defined Networks SOFTNETWORKING 2015, Barcelona, Spain, http://www.iaria.org/conferences2015/SOFTNETWORKING.html, [retrieved: 1, 2016]

[2] B. N. Astuto, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks", Communications Surveys and Tutorials, IEEE Communications Society, (IEEE), 2014, 16 (3), pp. 1617 – 1634.

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On Scalability of Software-Defined Networking", IEEE Comm. Magazine, February 2013, pp. 136-141.

[4] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A Flexible OpenFlow-Controller Benchmark," in European Workshop on Software Defined Networks (EWSDN), Darmstadt, Germany, October 2012.

[5] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow" in Proc. INM/WREN, 2010.

[6] T. Koponen, et. al., "Onix: a distributed control platform for large-scale production networks," in Proc. OSDI, 2010.

[7] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," Proc. HotSDN '12 Wksp., 2012.

[8] H. Yan-nan, W. Wen-dong, G. Xiang-yang, Q. Xi-rong, and C. Shi-duan, "On the placement of controllers in software-defined networks", ELSEVIER, Science Direct, vol. 19, Suppl.2, October 2012, pp. 92–97, http://www.sciencedirect.com/science/article/pii/S100588851160438X, [retrieved: 1, 2016].

[9] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proc. HotSDN, 2012, pp. 7–12.

[10] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization". Lecture Notes in Economics and Mathematical Systems, vol. 177. Springer-Verlag, pp. 468–486.

[11] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks," in ITC, Shanghai, China, 2013.

[12] Internet2 open science, scholarship and services exchange. http://www.internet2.edu/network/ose/, [retrieved: 1, 2016].

[13] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," IEEE JSAC, vol. 29, no. 9, 2011.

[14] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," in GLOBECOM 2011, 2011.

[15] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability aware controller placement for software-defined networks," in Proc. IM. IEEE, 2013, pp. 672–675.

[16] L. Muller, R. Oliveira, M. Luizelli, L. Gaspary, and M. Barcellos, "Survivor: an Enhanced Controller Placement Strategy for Improving SDN Survivability", IEEE Global Comm. Conference (GLOBECOM); 12/2014.

[17] D. Hochba "Approximation algorithms for np-hard problems", ACM SIGACT News, 28(2), 1997, pp. 40–52.

[18] https://www.python.org/doc/essays/blurb/, [retrieved: 1, 2016].

 ISBN: 978-1-61208-450-3

Controller Placement Problem to Enhance Performance in Multi-domain SDN Networks

Hidenobu Aoki, Norihiko Shinomiya
Graduate School of Engineering, Soka University, Tokyo, Japan
Emails: aoki39h@gmail.com, shinomi@soka.ac.jp

***Abstract*—As the use of Software-Defined Networking is extended, the deployment of multiple controllers has been required to deal with scalability and reliability issues. In such an SDN network, a network can be partitioned into sub-networks as controller's domains. In this case, there are mainly two issues to consider: 1) how to partition a network and 2) where to place controllers. Our previous work has concentrated on the network partitioning to satisfy several aspects in Software-Defined Networking. As an idea of our future work, this paper discusses and formulates the controller placement problem in multi-domain networks on the basis of network partitioning in our previous work.**

Keywords*—*Software-Defined Networking; multiple controllers; network partitioning; controller placement; graph theory.

I. INTRODUCTION

Software-Defined Networking (SDN) has been emerging as a new networking paradigm. The fundamental concept of SDN is to achieve programmable networking by separating the control and the data planes in individual network devices, such as switches and routers. As the use of SDN is extended, SDN networks with multiple controllers have been proposed to deal with scalability and reliability issues [1].

In such an SDN network with multiple controllers, a network can be divided into sub-networks, and controllers are in charge of one of them as their administrative domains. This could reduce the overall complexity of the whole network management and the computational load of each controller as well as handling flow setup requests faster and more efficiently. Furthermore, when a controller failure occurs, it could alleviate spreading its negative effects to the rest of the network [2].

Here, there would be two major aspects to consider for such a network:

1) How to partition a network to decide controller domains
2) Where to locate controllers in each domain.

Because network partitioning determines network resources and topology distribution, it could affect the various aspects of network performance, such as controller load balance and reliability [3]. This issue is referred to as *the network partitioning problem* in our previous work. On the other hand, the selection of controller locations has been discussed in many research papers as it has various influences on communications between switches and a controller in terms of latency, stability, or efficiency. Generally, this issue is called *the controller placement problem*.

Our previous work has mainly focused on network partitioning methods based on graph clustering [4]. Thus, aiming at more effective management of multi-domain SDN networks, this paper addresses the controller placement problem based on the network partitioning as an extension of our previous work.

The rest of the paper is organized as follows. Section II describes the related work of the network partitioning and controller placement. Section Ⅲ presents the model and problem formulation. Section Ⅳ addresses the conclusion and our future work.

II. RELATED WORK

A. Network Partitioning

As mentioned earlier, network partitioning could be related to the various aspects of network operations and performances of SDN.

For instance, the load balance among controllers would be one of the major issues. The controller load is mainly network provisioning and status collection overhead within a domain [5]. Hence, simply, the more switches a controller needs to manage, the heavier load the controller has to bear.

Moreover, in case of a link failure, it would be preferable that a network topology in a domain ensures redundancy so that the failure can be recovered within the domain; otherwise, additional inter-controller communication would be required.

In addition, from the perspective of traffic load balancing, switches in each domain might need to have multiple paths to controllers and other switches to distribute data traffic.

Considering those issues above, our previous work has proposed network partitioning methods based on the concept of conductance and cycle structures in networks [4][6].

B. Controller Placement

After the first proposal by Heller et al in [7], the controller placement problem has been studied in many researches. The most common objective for the problem is to minimize controller-switch latency [8][9]. Particularly, it has been widely studied to locate a controller based on the closeness to switches.

In addition to controller-switch distance, survivability is also the other metric for the controller deployment [10][11]. In this type of metric, the number of disjoint paths, which do not share links among them, between switches and a controller is maximized to ensure logical connections between them in case of link failures in SDN in-band model.

 ISBN: 978-1-61208-450-3

As stated above, the controller placement problem and its metrics have been well studied. However, the controller placement in partitioned networks has not been discussed in detail. By using those existing metrics, this paper formulates the controller placement problem based on the network partitioning in our previous work.

III. Problem Formulation

This section describes the graph definitions and the metrics for the control placement.

A. Graph Definitions

For a network graph $G = (V, E)$, a set of nodes $V = \{v_i\}$ denotes network devices, such as routers and switches, and a set of edges $E = \{e_k\}$ represents links between those devices. Considering the network partitioning, local domains of controllers and its set are defined as follows:

$$G_1^l = (V_1^l, E_1^l), G_2^l = (V_2^l, E_2^l), \ldots, G_k^l = (V_k^l, E_k^l), \quad (1)$$

$$\mathbf{G^1} = \{G_1^l, G_2^l, ..., G_j^l ..., G_k^l\}. \quad (2)$$

B. Evaluation Metrics for Controller Placement

The controller placement problem in multi-domain networks can be formulated as an optimization problem to find an appropriate node c_j for a controller location in each domain where a given evaluation function is optimized. Here, assume that domains are given by network partitioning, and a set of controller locations in domains is denoted as

$$C = \{c_1, c_2, \ldots, c_j, \ldots, c_k\}. \quad (3)$$

1) Controller-switch latency: The most commonly used metric is the longest distance between a switch $v_i \in V_j^l$ and a controller $c_j \in C$, which represents the worst controller-switch latency. Supposing that the distance is the shortest path length between them denoted as $dist(v_i, c_j)$, the longest distance between a switch and controller in a domain is defined as

$$L(G_j^l) = \max_{v_i \in V_i^l \setminus \{c_j\}} dist(v_i, c_j). \quad (4)$$

Thus, for a network as a whole, the task is to find a set of controller locations C to satisfy the following function:

$$\text{Minimize} \sum_{G_j^l \in \mathbf{G^1}} L(G_j^l). \quad (5)$$

2) Survivability: In the controller placement problem, survivability is denoted as the number of edge disjoint paths between switches and a controller. By representing the number of edge disjoint paths between a switch v_i and a controller c_j of a domain G_j^l as δ_{ij}, the survivability of a domain is denoted as

$$S(G_j^l) = \frac{\sum_{v_i \in G_j^l \setminus \{c_j\}} \delta_{ij}}{|V| - 1}. \quad (6)$$

Then, the evaluation function is described as

$$\text{Maximize} \sum_{G_j^l \in \mathbf{G^1}} S(G_j^l). \quad (7)$$

Therefore, the whole algorithm will include two phases: 1) partitioning a network and 2) finding the locations of controllers in individual partitioned networks so as to satisfy the metrics.

IV. Conclusion and Future Work

In multi-domain SDN networks, two issues have been discussed: 1) Network partitioning and 2) Controller placement. This paper has formulated the controller placement problem for multi-domain networks, which considers the network partitioning in our previous work.

As the first step in future tasks, the generally defined metrics of the controller placement, such as controller-switch latency and survivability will be examined on the partitioned networks.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 26330120.

References

[1] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," Communications Magazine, IEEE, vol. 51, no. 2, February 2013, pp. 136–141.

[2] H. Xie, T. Tsou, D. Lopez, H. Yin, and V. Gurbani, "Use cases for alto with software defined networks," Working Draft, IETF Secretariat, Internet-Draft draft-xie-alto-sdn-extension-use-cases-01. txt, 2012.

[3] X. Li, P. Djukic, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014, pp. 1–8.

[4] H. Aoki and N. Shinomiya, "Network partitioning problem for effective management of multi-domain sdn networks," vol. 3&8, no. 8. IARIA, 2015, pp. 171–181.

[5] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in sdn-based core networks," in Teletraffic Congress (ITC), 2013 25th International. IEEE, 2013, pp. 1–9.

[6] J. Nagano and N. Shinomiya, "Efficient information sharing among distributed controllers of openflow network with bi-connectivity," in Computing, Networking and Communications (ICNC), 2015 International Conference on. IEEE, 2015, pp. 320–324.

[7] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 7–12.

[8] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," Communications Letters, IEEE, vol. 18, no. 8, 2014, pp. 1339–1342.

[9] E. Borcoci, R. Badea, S. G. Obreja, and M. Vochin, "On multi-controller placement optimization in software defined networking-based wans," ICN 2015, 2015, p. 273.

[10] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," Communications, China, vol. 11, no. 2, 2014, pp. 38–54.

[11] L. F. Muller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving sdn survivability," in Global Communications Conference (GLOBECOM), 2014 IEEE. IEEE, 2014, pp. 1909–1915.

 ISBN: 978-1-61208-450-3

Performance Comparison of SDN Solutions for Switching Dedicated Long-Haul Connections

Nageswara S. V. Rao

Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA
Email: raons@ornl.gov

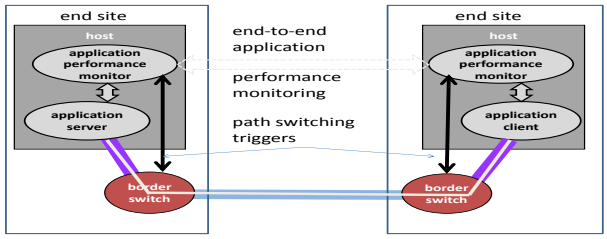
***Abstract*—We consider scenarios with two sites connected over a dedicated, long-haul connection that must quickly fail-over in response to degradations in host-to-host application performance. We present two methods for path fail-over using OpenFlow-enabled switches: (a) a light-weight method that utilizes host scripts to monitor the application performance and dpctl API for switching, and (b) a generic method that uses two OpenDaylight (ODL) controllers and REST interfaces. The restoration dynamics of the application contain significant statistical variations due to the controllers, north interfaces and switches; in addition, the variety of vendor implementations further complicates the choice between different solutions. We present the impulse-response method to estimate the regressions of performance parameters, which enables a rigorous and objective comparison of different solutions. We describe testing results of the two methods, using TCP throughput and connection rtt as main parameters, over a testbed consisting of HP and Cisco switches connected over long-haul connections emulated in hardware by ANUE devices. The combination of analytical and experimental results demonstrates that dpctl method responds seconds faster than ODL method on average, while both methods restore TCP throughput.**

Keywords–Software defined networks; OpenFlow; Opendaylight; controller; long-haul connection; impulse-response; testbed.

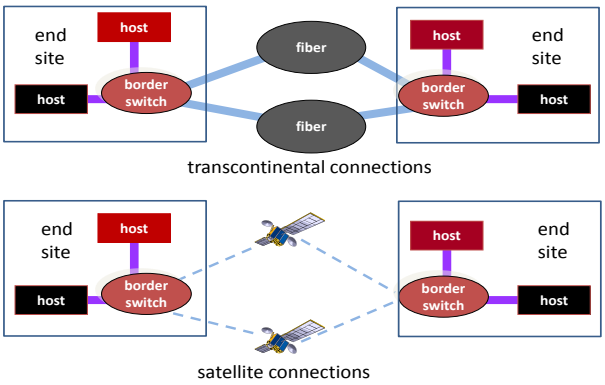
I. INTRODUCTION

We consider scenarios with two sites connected over dedicated long-haul connections such as transcontinental fibers or satellite links, as illustrated in Figure 1. Different client-server application pairs are executed at different times on host systems at the sites, to support data transfers, on-line instrument monitoring, and messaging. The connection quality can degrade due to a variety of factors such as equipment failures, weather conditions, and geographical events, which in turn affect the host-to-host application performance. As a mitigation strategy, a physically diverse and functionally equivalent *standby* path is switched to when the application performance degrades.

The performances of application pairs are monitored on host systems, and the current *primary* path is switched out when needed, for example, by modifying Virtual Local Area Networks (VLAN) and route tables on border switches and routers, respectively. In our use cases, human operators watch the host-level performance monitors, and invoke Command Line Interface (CLI) commands or web-based interfaces of network devices for path switching. Since triggers for path switching are dynamically generated by application pairs, they are not adequately handled by conventional hot-standby layer-2/3 solutions that solely utilize connection parameters. For example, certain losses may be tolerated by messaging applications but not by monitoring and control applications of instruments and sensors. Currently, the design and operation of such application-driven fail-over schemes require a detailed knowledge of host codes, such as Linux scripts, and the specialized interfaces and languages of switches, such as CLI and custom TL1 and CURL scripts (which currently vary significantly among the vendor products). Furthermore, fail-over operations must be coordinated between two physically-separated operations centers located at the end sites.



(a) host-to-host application pairs



(b) two connection modalities

Figure 1. Two sites connected over long-haul connections.

A. SDN Solutions

The rapidly evolving *Software Defined Networks* (SDN) technologies [1], [2] seem particularly well-suited for automating the path switching tasks, when combined with host monitoring codes. In particular, SDN technologies provide two distinct advantages:

(a) trigger modules of new applications can be "dropped in place" since they use generic northbound interfaces, and
(b) switches from different vendors with virtual interfaces can be simply swapped, avoiding the re-work needed for custom interfaces and network operating systems.

 ISBN: 978-1-61208-450-3

The execution of this seemingly direct approach, surprisingly however, requires comparing and selecting from among rather complex configurations. Due to the rapid development of SDN technologies [1], there is a wide array of options for controllers and switches, which in turn leads to a large number of their solution combinations [3]. Indeed, their complexity and variety requires systematic analysis and comparison methods to assess their operational effectiveness and performance [4], such as recovery times. In addition, compared to certain data-center and network provisioning scenarios for which SDN technologies have been successfully applied [5], [6], these long-haul scenarios present additional challenges that require newer solutions: (a) single controller solutions that have been extensively used in several applications [1] are not practical for managing the border switches at end sites due to the large latency, and (b) solutions that require a separate control-plane infrastructure, such as DISCO [7], are cost prohibitive since they require additional control plane connections.

B. Outline of Contributions

In this paper, we present automated software solutions for path fail-over by utilizing two controllers, one at each site, that are coordinated over a single connection through measurements. We first describe a light-weight, custom designed *dpctl method* for OpenFlow border switches that uses host Linux bash scripts. This script is about a hundred lines of code, which makes it easier to analyze for its performance and security aspects. We then present a more generic *ODL method* that utilizes two OpenDaylight (ODL) controllers [8] located at the end sites. The executional path of this approach is more complex compared to the dpctl method since it involves communications using both northbound and southbound ODL interfaces and invoking several computing modules within ODL software stack. Thus, a complete performance and security analysis of this method requires a closer examination of a much larger code base that includes both host scripts and corresponding ODL modules, including its embedded http server [9].

We present implementation and experimental results using a testbed consisting of Linux hosts, HP and Cisco border switches, and ANUE long-haul hardware connection emulation devices. We utilize TCP throughput as a primary performance measure for the client-server applications, which is affected by the connection rtt and jitter possibly caused by path switching, and the available path capacity. Experimental results show that both dpctl and ODL methods restore the host-to-host TCP throughput within seconds by switching to the standby connection after the current connection's RTT is degraded (due to external factors). However, the restoration dynamics of TCP throughput show significant statistical variations, primarily as a result of the interactions between the path switching dynamics of the controllers and switches, and the highly non-linear dynamics of TCP congestion control mechanisms [10]–[12]. As a result, direct comparisons of individual TCP throughput traces corresponding to fail-over events are not very instructive in reflecting the overall performance of the two methods.

To objectively compare the performances [4] of these two rather dissimilar methods, we propose the *impulse-response* method that captures the average performance by utilizing measurements collected in response to a train of path degradation events induced externally. We establish a statistical basis

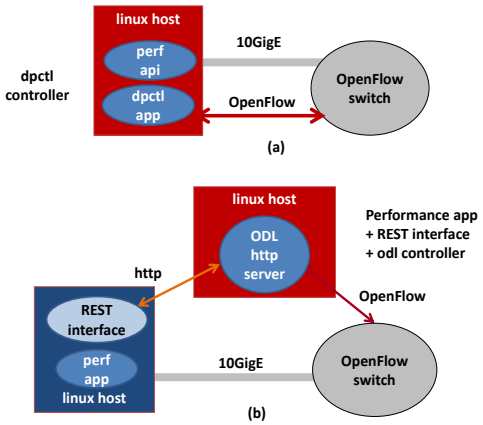


Figure 2. Configurations of dpctl and ODL controllers.

for this method using the finite-sample theory [13] by exploiting the underlying monotonic properties of the performance parameters during the degradation and recovery periods. This analysis enables us to objectively conclude that on average the dpctl method restores the TCP throughput several seconds faster than ODL method for these scenarios.

The organization of this paper is as follows. A coordinated controllers approach using dpctl and ODL methods is described in Section II. An experimental testbed is described in Section III-A, and the results of experiments using dpctl and ODL methods using five different configurations are presented in Section III-B. The impulse response method and its statistical basis are presented in Sections IV-A and IV-B, respectively. Conclusions are presented in Section V.

II. Coordinated SDN Controllers

For these long-haul scenarios, single controllers solutions or solutions requiring separate control plane are not effective, although such approaches with stable control-plane connections have been effective in path/flow switching over local area networks using OpenFlow [5], [14] and cross-country networks using customized methods [6], [7], [15]. Since the controller has to be located at a site, when the primary connection degrades, it may not be able to communicate effectively with the remote site. Our approach is to utilize two controllers, one at each site, which are "indirectly" coordinated based on the monitored application-level performance parameters. When path degradation is inferred by a host script, the controller at that site switches to the fail-over path by installing the appropriate flow entries on its border switch. If the primary path degrades, for example, resulting in increased latency or loss rate, its effect is typically detected at both hosts by the monitors, and both border switches fail-over to the standby path at approximately the same time. If the border switch at one site fails-over first, the connection loss will be detected at the other site which in turn triggers the fail-over at the second site. Also, one-way failures lead to path switching at one site first, which will be seen as a connection loss at the other site, thereby leading to path switching at that site as well. Due to recent developments in the SDN technologies, both in open software areas [16] and specific vendor implementations [17], [18], there are many different ways such a solution can be implemented. We only consider OpenFlow solutions which are implemented using open standards and software [5].

 ISBN: 978-1-61208-450-3

A. *dpctl Method*

As a part of OpenFlow implementation, some vendors support dpctl API which enables hosts to communicate with switches to query the status of flows, insert new flows and delete existing flows. It has been a very useful tool primarily for diagnosing flow implementations by using simple host scripts; however, some vendors such as Cisco do not provide dpctl support. We utilize dpctl API in a light-weight host script that constantly monitors the connection rtt and detects when it crosses a threshold and invokes dpctl to implement the fail-over as shown in Figure 2(a). The OpenFlow entries for switching to the standby path are communicated to the switch upon the detection of connection degradation. This script consists of under one hundred lines of code and is flexible in that the current connection monitoring module can be replaced by another one such as TCP throughput monitor using iperf. Compared to methods that use separate OpenFlow controllers, this method compresses both performance monitoring and controller modules into one script, and thereby avoids the northbound interface altogether; for ease of reference, we refer to this host code as the *dpctl controller*.

B. *OpenDaylight Method*

We now utilize two ODL Hydrogen controllers and REST interfaces to implement fail-over functionality using OpenFlow flows as shown in Figure 2(b). ODL is an open source controller [8] that communicates with OpenFlow switches, and is used to query, install and delete flow entries on them using its southbound interface. The applications communicate with ODL controller via the northbound interface to query, install and delete flows. ODL software in our case runs on Linux workstations called the controller workstations, and the application monitoring codes can be executed on the same workstation in the *local* mode or on a different workstation in the *remote* mode.

The same performance monitoring codes of the dpctl method above are used in this case to detect path degradations but are enhanced to invoke python code to communicate new flows for switching paths to ODL controllers via REST interfaces; the content of these flow entries are identical to the previous case. Thus, both the software and executional paths of this method are much more complicated compared to previous case, and also the ODL controllers are required to run constantly on the servers at end sites. Also, this code is much more complex to analyze since it involves not only the REST scripts but also the ODL stack which by itself is a fairly complex software. The executional path is more complex since it involves additional communication over both northbound and southbound interfaces of ODL controllers.

III. Experimental Results

In this section, we first describe the testbed and then describe the experimental results.

A. *Emulation Testbed*

The experimental testbed consists of two site LANs, each consisting of multiple hosts connected via 10GigE NICs to the site's border switch. The border switches are connected to each other via a local fiber connection of a few meters in length, and also via two ANUE devices that emulate long-haul connections in hardware, as shown in Figure 3. Tests use pairs

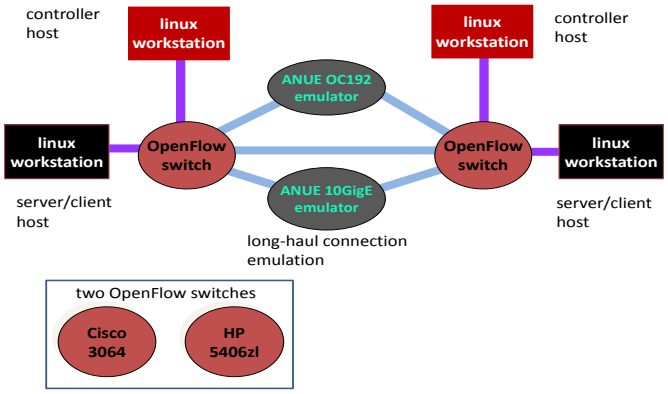


Figure 3. Testbed of two sites connected over local and emulated connections.

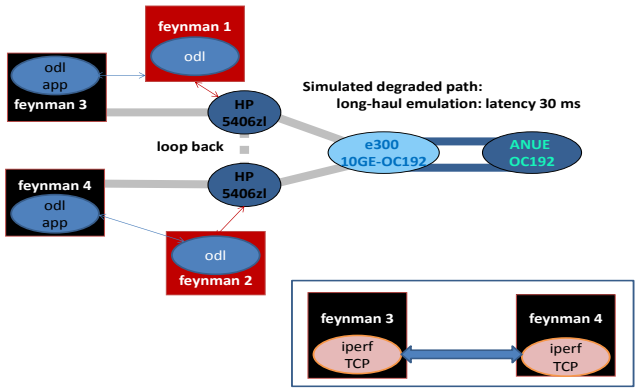
of HP 5064zl and Cisco 3064 devices as border switches, both of which are OpenFlow-enabled but only HP switches support dpctl. The OC192 ANUE devices emulate connections with rtts in the range of [0-800] milliseconds with a peak capacity of 9.6 Gbps. The conversion between 10GigE LAN packets from the border switches and long-haul OC192 ANUE packets is implemented using a Force10 E300 switch, as shown in Figure 4. ANUE devices are utilized primarily to emulate the latencies of long-haul connections, both transcontinental fiber and satellite, to highlight the overall recovery dynamics.

Two classes of Linux hosts are connected to the border switches. The controller hosts (feynman1 and feynman2) are utilized to execute ODL controllers, and client and server hosts (feynman3 and feynman4) are used to execute monitoring and trigger codes along with client server codes, for example, iperf clients and servers. Five different configurations are employed in the tests as shown in Table I. The *dpctl* tests utilize only the client and server hosts to execute both monitoring and switching codes. Configuration A corresponds to these tests with the monitoring and dpctl scripts running on server/client hosts, and it uses HP border switches. For *ODL remote mode* tests, the monitoring codes on client/server hosts utilize REST interface to communicate flow message needed for fail-over; both HP and Cisco switches are used in these tests. Configurations C-E implement these tests, which employ ODL controllers running on controller hosts and monitoring scripts running on server/client hosts. In *ODL local mode* tests, the monitoring and client/server codes are executed directly on the control hosts. Configuration B implements these tests, and it is identical to Configuration C except its scripts are executed on controller hosts. The measurements in Configurations B and C are quite similar, and hence we mostly present the results of the latter.

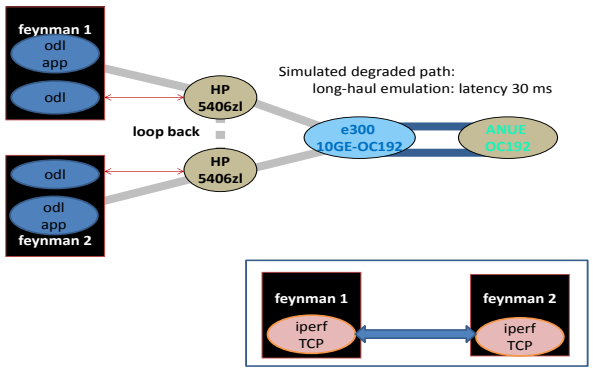
In our experiments, connection degradation events are implemented by external codes using two different methods:

(a) *Path switching using dpctl:* The current physical path with a smaller rtt is switched to a longer emulated path, whose rtt is sufficiently long to trigger the fail-over. This switching is accomplished by using dpctl codes that install OpenFlow entries on the border switches to divert the flow from current path to the longer path. The packets enroute on the current path will be simply dropped and the short-

 ISBN: 978-1-61208-450-3



(a) Configurations C-E: remote



(b) Configuration B: local

Figure 4. Remote and local modes of ODL controller configurations.

term TCP throughput becomes zero. After the fail-over, the path is switched back to the original path, and the TCP flow recovers gradually back to previous levels.

(b) *RTT extension using curl scripts:* The current connection's rtt is increased by changing the setting on ANUE device to a value above the threshold to trigger the fail-over. This is accomplished by using curl scripts that access ANUE http interface. Unlike the previous case, the packets enroute on the current path are not dropped but are delayed; thus, the instantaneous TCP throughput does not always become zero but is reduced significantly. After the fail-over, the original rtt is restored, and TCP throughput recovers gradually to previous levels.

The first degradation method using dpctl to switch the paths is only implemented for configurations with HP border switches in Configurations A - C. The second method is used for both HP and Cisco systems in Configurations D and E, and since the curl scripts are used here to change delay settings on ANUE devices, the border switches are not accessed.

B. Controller Performance

TCP throughput measurements of the connection are constantly monitored using iperf, and the Linux hosts use the default CUBIC congestion control modules [19]. The rtt between end hosts is also constantly monitored using ping, and

test configuration	controller method	path degradation	switch vendor
A	dpctl	path switch	HP
B	ODL local	path switch	HP
C	ODL remote	path switch	HP
D	ODL remote	rtt extension	HP
E	ODL remote	rtt extension	Cisco

Table I. Five test configurations with two controllers, two connection degradation methods and two switch vendors.

path switching is triggered when it crosses a set threshold. The path degradations are implemented as periodic impulses and the responses are assessed using the recovery profiles of TCP throughput captured at one second intervals. Also, the ANUE dynamics in extending the rtt affect the TCP throughput recovery, and we obtain additional baseline measurements by utilizing a direct fiber connection that avoids packets being routed through ANUE devices. Thus, TCP throughout traces in our tests capture the performances of: (a) controllers, namely, dpctl and ODL, in responding to fail-over triggers from monitoring codes, and in modifying the flow entries on switches, typically by deleting the current flows and inserting the ones for the standby path, and (b) border switches in modifying the flows entries in response to controller messages and re-routing the packets as per new flow entries.

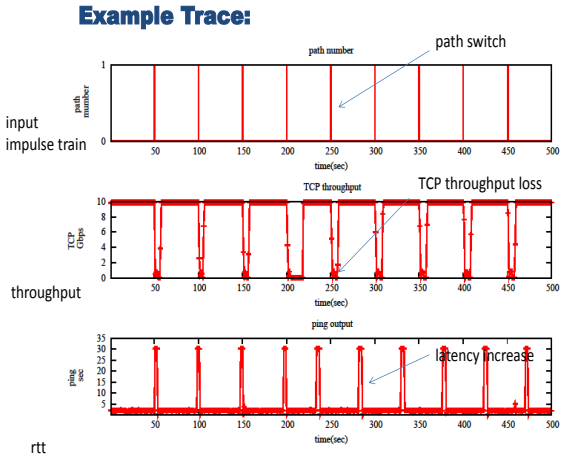


Figure 5. Trace of impulse response of TCP throughput for dpctl method with local primary path and path switching degradation.

An example TCP throughput trace of a test run of Configuration A is shown in Figure 5 for the dpctl method, with fiber connection as the primary path, and using the path switching degradation method. The connection rtt is degraded at a periodicity of 50 seconds by externally switching to the longer ANUE path, and the change is detected, as shown in the bottom plot, which in turn triggers the fail-over. Three different TCP recovery profiles from the tests are shown in Figure 6: (a) *Configuration A*: dpctl method using HP switches with connection degradation by path switching, (b) *Configuration D*: ODL method using HP switches with connection degradation by rtt extension, and (c) *Configuration E*: ODL method using Cisco switches with connection degradation by rtt extension. As seen in these plots, TCP response dynamics contain significant variations for different degradation events of the same configuration as well as between different configurations.

 ISBN: 978-1-61208-450-3

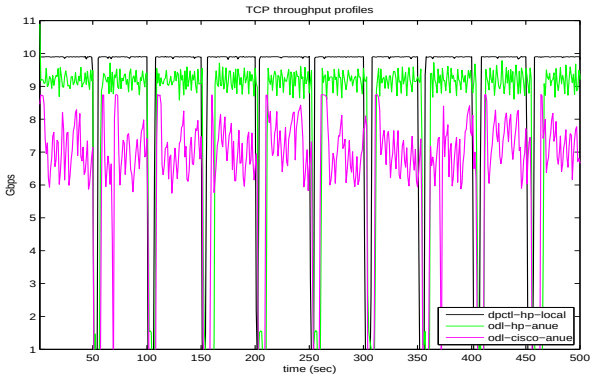


Figure 6. TCP throughput for dpctl method for configuration A and ODL methods for configuration D and E.

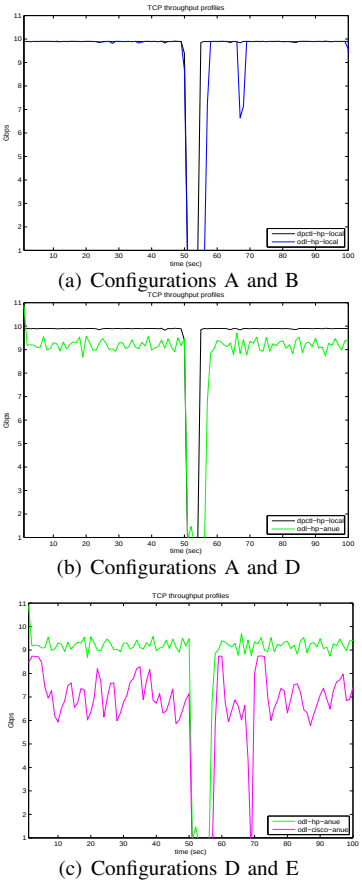
The individual TCP throughput recovery responses to single path degradation events reveal more details of the difference between the configurations as shown in Figure 7. The delayed response of ODL method compared to dpctl method can be seen in Figure 7(a) for Configurations A and B. Since the packets in transit during the switching are simply lost during path switching, the instantaneous TCP throughput rapidly drops to zero for Configuration A. On the other hand, some of the packets in transit when rrt is extended are delivered, and as a result TCP throughput may be non-zero in some cases, as shown in Figure 7(b). Another aspect is that, TCP throughput recovers to 10Gbps when the direct fiber connection is used between the switches, but only peaks around 9 Gbps when packets are sent via ANUE connection with zero delay setting as shown in Figure 7(b). Also, the recovery profiles are different between HP and Cisco switches in otherwise identical Configurations D and E as shown in Figure 7(c). Thus, TCP dynamics depend both on the controller in terms of recovery times, and on the switches in terms of peak throughput achieved and its temporal stability.

C. Switch Performance

TCP performance is effected by the path traversed by the packets between the border switches, in addition to its dependence on dpctl and ODL methods as described in the previous section. In configurations A and B, the primary connection is a few meters of fiber between the switches, and TCP throughput is restored to around 10 Gbps after the fail-over as shown in Figure 7(a). In Configurations D and E, the packets are sent through the emulated connection OC192 ANUE emulator with a peak capacity of 9.6 Gbps, and both peak value and the dynamics of TCP throughput are affected as shown in Figure 7(b). Furthermore, the connection modality affects HP and Cisco switches differently as shown in Figure 7(c) in that the latter reach somewhat lower peak throughput and exhibit larger fluctuations.

IV. Impulse Response Method

We present the *impulse response method* in this section that captures the overall recovery response by "aggregating" generic (scalar) performance measurements (such as TCP throughput as in previous section) collected in response to periodic connection degradations.



(a) Configurations A and B

(b) Configurations A and D

(c) Configurations D and E

Figure 7. Trace of impulse response of TCP throughput for dpctl method with local primary path and path switching degradation.

A. Response Regression Function

A configuration X that implements the fail-over is specified by its controllers and switches that implement fail-over, and also the monitoring and detection modules that trigger it. Let $\delta(t - iT), i = 0, 1, \ldots, n$, denote the input impulse train that degrades the connection at times $iT + T_D$, where t represents time, T is the period between degradations and $T_D < T$ is the time of degradation event within the period. Let $T^X(t)$ denote the parameter of interest, such as TCP throughput, corresponding to $\delta(t-iT), i = 0, 1, \ldots, n$ as shown in Figure 6 for configurations X=A,D,E. Let $R^X(t) = \mathbf{B} - T^X(t)$ denote the response that captures the "unrealized" portion of peak performance level $\mathbf{B}$, for example, the residual bandwidth of a connection with capacity $\mathbf{B}$. We define the *impulse response function* $R^X(t)$ such that

$$R_i^X(t) = R^X(t - iT), t \in [0, T)$$

is the response to ith degradation event $\delta(t - iT), i = 0, 1, \ldots, n$. An ideal impulse response function is also an impulse train that matches the input, wherein each impulse represents the instantaneous degradation detection, fail-over and complete recovery. But in practice, each $R_i^X(t)$ is a "flattened" impulse function whose shape is indicative of the effectiveness of fail-over. In particular, its leading edge represents the effect of degradation and its trailing edge represents that of recovery, and the narrower this function is the quicker is the recovery.

 ISBN: 978-1-61208-450-3

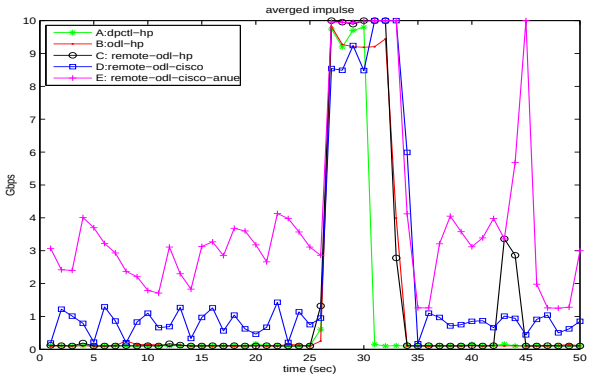


Figure 8. Examples of impulse response functions for Configurations A-E for a single path degradation.

Examples of $R_1^X(.)$ are shown Figure 8 for configurations A-E; these TCP measurements show significant temporal variations that persist across the different degradation events, which make it difficult to objectively compare these single-event time plots.

We define the *response regression* of configuration X as

$$\bar{R}^X(t) = E\left[R_i^X(t)\right] = \int R_i^X(t) d\mathbf{P}_{R_i^X(t)},$$

for $t \in [0, T)$, where the underlying distribution $\mathbf{P}_{R_i^X(t)}$ is quite complex in general since it depends on the dynamics of controllers, switches, end hosts, application stack and monitoring and detection modules that constitute X. It exhibits an overall decreasing profile for $t \in [0, T_D + T_I]$ followed by an increasing profile for $t \in (T_D + T_I, T]$, where T_I is the time needed for the application to react to connection degradation. After the fail-over, TCP measurements exhibit an overall increasing throughput until it reaches its peak as it recovers after becoming nearly zero following the degradation. We consider that a similar overall behavior is exhibited by the general performance parameters of interest.

The *response mean* $\hat{R}_i(t)$ of $\bar{R}_i(t)$ based on discrete measurements at times $t = j\delta$, $j = 0, 1, ..., T/\delta$, is

$$\hat{R}^X(j\delta) = \frac{1}{n}\sum_{i=1}^{n}\left(R_i^X(j\delta)\right)$$

which captures the average profile. Examples of $\hat{R}_i^X(.)$ for TCP throughput are shown Figure 9 for different configurations based on 10 path degradations with $T = 50$ seconds between them, which show the following general trends.

- The dpctl method responds seconds faster than ODL method as indicated by its sharper shape, although their leading edges are aligned as shown in Figure 9(a).
- The connection degradation implemented by the rtt extension has a delayed effect on reducing the throughput compared to the path switching degradation method as indicated by its delayed leading edge in Figure 9(b).
- The dynamic response of regression profiles of HP 5604zl and Cisco 3064 switches are qualitatively quite similar as shown in Figure 9(c), but the latter achieved somewhat lower peak throughput overall.

In view of the faster response of dpctl method, we collected additional measurements in configurations A and C using

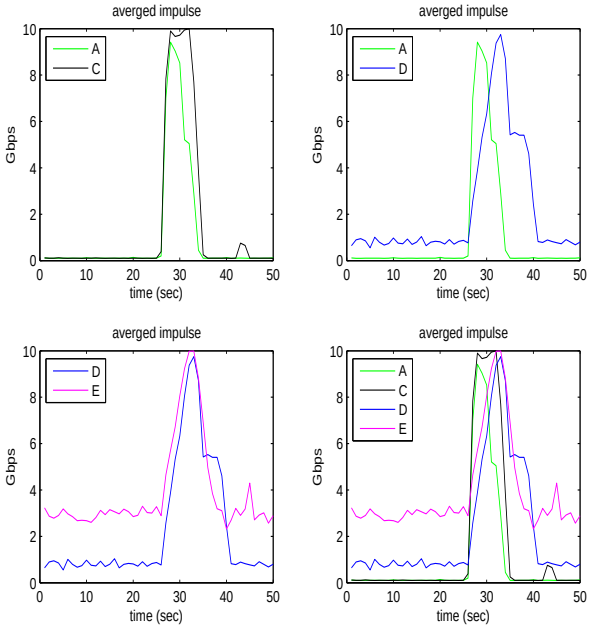
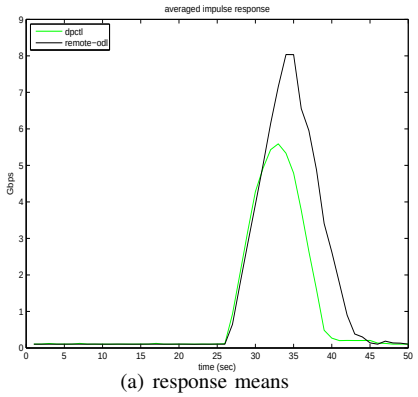
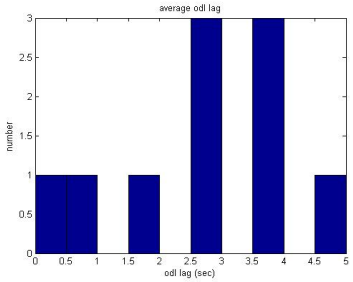


Figure 9. Impulse response regressions for $n = 10$ and $T = 50$ sec: (a) top-left: A and C, (b) top-right: A and D, (c) bottom-left: D and E , and (d) bottom-right: all.



(a) response means



(b) delay histogram

Figure 10. Comparison of response means of dpctl (configuration A) and ODL (configuration C) methods using 100 path degradations with $T = 50$ seconds.

 ISBN: 978-1-61208-450-3

100 path degradations, and the resultant response means are somewhat smoother compared to 10 degradations as shown in Figure 10 (a) for both dpctl and ODL methods. Furthermore, the response mean of ODL method remains consistent with more measurements, and a histogram of the relative delays of ODL method compared to dpctl method is plotted in Figure 10(b), and they are in the range of 2-3 seconds in the majority of cases. These measurements clearly show the faster response of the dpctl method for these scenarios.

B. Statistical Analysis

A generic empirical estimate $\tilde{R}^X(t)$ of $\bar{R}(t)$ based on measurements at times $t = j\delta$, $j = 0, 1, ..., T/\delta$, is

$$\tilde{R}^X(j\delta) = \frac{1}{n}\sum_{i=1}^{n}\left[g\left(R_i^X(j\delta)\right)\right]$$

for an estimator function g. We consider that the function class $\mathcal{M}$ of $\tilde{R}^X(.)$ consists of unimodal functions, each of which consists of degradation and recovery parts when viewed as a function of time. For ease of notation, we also denote $\tilde{R}^X(.)$ by f in this section such that it is composed of a degradation function f_D and a recovery function f_R as follows:

$$f(R_i(t)) = \begin{cases} f_D(R_i(t)) & \text{if } t \in [0, T_D + T_I] \\ f_R(R_i(t)) & \text{if } t \in (T_D + T_I, T] \end{cases} \quad (1)$$

where $f_D \in \mathcal{M}_D$ and $f_R \in \mathcal{M}_R$ correspond to the leading and trailing edges of the response regression. The *expected error* $I(f)$ of the estimator f is given by

$$\begin{aligned} I(f) &= \int [f(t) - R_i^X(t)]^2 d\mathbf{P}_{R_i^X(t),t} \\ &= \int_{[0,T_D+T_I]} [f_D(t) - R_i^X(t)]^2 d\mathbf{P}_{R_i^X(t),t} \\ &\quad + \int_{(T_D+T_I,T]} [f_D(t) - R_i^X(t)]^2 d\mathbf{P}_{R_i^X(t),t} \\ &= I_D(f_D) + I_R(f_R). \end{aligned}$$

The *best expected estimator* $f^* = (f_D^*, f_R^*) \in \mathcal{M}$ minimizes the expected error $I(.)$, that is

$$I(f^*) = \min_{f \in \mathcal{M}} I(f).$$

The *empirical error* of an estimator f is given by

$$\hat{I}(f) = \frac{\delta}{Tn}\sum_{i=1}^{n}\sum_{j=1}^{T/\delta}\left[f(j\delta) - \left(R_i^X(j\delta)\right)\right]^2.$$

The *best empirical estimator* $\hat{f} = (\hat{f}_D, \hat{f}_R) \in \mathcal{M}$ minimizes the empirical error $\hat{I}(.)$, that is,

$$\hat{I}(\hat{f}) = \min_{f \in \mathcal{M}} \hat{I}(f).$$

Since the response mean $\hat{R}(t)$ is the mean at each observation time $j\delta$, it achieves zero mean error, which in turn leads to zero empirical error, that is, $\hat{I}\left(\hat{R}\right) = 0$; thus, it is a best empirical estimator. By ignoring the minor variations for the smaller values of n, we assume that $\hat{R}$ is composed of a non-decreasing function $\hat{R}_D$ followed by a non-increasing function $\hat{R}_R$ that correspond to decreasing and increasing parts of the performance parameter (such as TCP throughput), respectively. This assumption is valid for the response means of dpctl and ODL methods in Configurations A and C, respectively, shown in Figure 10. In both cases, the response mean is composed of an increasing part followed by a decreasing part once the small variations in the tail of ODL method are ignored.

We will now show that Vapnik-Chervonenkis theory [13] guarantees that the response mean $\hat{R}(t)$ is a good approximation of the response regression $\bar{R}(t)$, and furthermore its performance improves with more measurements from connection degradation events. Such performance guarantee is a direct consequence of the monotone nature of the underlying f_D and f_R functions. Furthermore, this performance guarantee is distribution-free, that is, independent of the underlying distributions due to controllers and switches, and is valid under very general conditions [13] on the variations of performance (such as TCP throughput) measurements. We now provide an outline of the proof of this result. Let $\hat{R} = \left(\hat{R}_D, \hat{R}_R\right)$ such that the estimator is decomposed into two monotone parts, namely non-decreasing $\hat{R}_D$ and non-increasing $\hat{R}_R$ such that $\hat{I}(\hat{R}) = \hat{I}_D(\hat{R}_D) + \hat{I}_R(\hat{R}_R)$. By using Vapnik-Chervonenkis theory [13] we have

$$\begin{aligned} &\mathbf{P}\left\{I\left(\hat{R}\right) - I(f^*) > \epsilon\right\} \\ &\leq \mathbf{P}\left\{I\left(\hat{R}_D\right) - I(f_D^*) > \epsilon/2\right\} \\ &\quad + \mathbf{P}\left\{I\left(\hat{R}_R\right) - I(f_R^*) > \epsilon/2\right\} \\ &\leq \mathbf{P}\left\{\max_{h \in \mathcal{M}_D} |I_D(h) - \hat{I}_D(h)| > \epsilon/4\right\} \\ &\quad + \mathbf{P}\left\{\max_{h \in \mathcal{M}_R} |I_R(h) - \hat{I}_R(h)| > \epsilon/4\right\} \\ &\leq 16\mathcal{N}_\infty\left(\frac{\epsilon}{2\mathbf{B}}, \mathcal{M}_D\right) n e^{-\epsilon^2 n/(8\mathbf{B})^2} \\ &\quad + 16\mathcal{N}_\infty\left(\frac{\epsilon}{2\mathbf{B}}, \mathcal{M}_R\right) n e^{-\epsilon^2 n/(8\mathbf{B})^2} \end{aligned}$$

where $\mathcal{N}_\infty(\epsilon, \mathcal{A})$ is the ϵ-cover of function class $\mathcal{A}$ under L_∞ norm. In the above derivation, the first inequality follows since the negation of the condition in either right term implies the negation of the condition in left term. The second inequality follows from the uniform convergence property applied to each term [13], and the third inequality follows by applying the uniform bound for each term ([20], p. 143). Due to the monotonicity of functions in $\mathcal{M}_D$ and $\mathcal{M}_R$, their total variation is upper bounded by $\mathbf{B}$, which provides us the following upper bound ([20], p. 175): for $\mathcal{A} = \mathcal{M}_D, \mathcal{M}_R$, we have

$$\mathcal{N}_\infty\left(\frac{\epsilon}{2\mathbf{B}}, \mathcal{A}\right) < 2\left(\frac{4n}{\epsilon^2}\right)^{(1+2\mathbf{B}/\epsilon)\log_2(2\epsilon/\mathbf{B})}.$$

By using this bound, we obtain

$$\begin{aligned} &\mathbf{P}\left\{I\left(\hat{R}_i\right) - I(f^*) > \epsilon\right\} \\ &< 64\left(\frac{4n}{\epsilon^2}\right)^{(1+2\mathbf{B}/\epsilon)\log_2(2\epsilon/\mathbf{B})} n e^{-\epsilon^2 n/(8\mathbf{B})^2}. \end{aligned}$$

The exponential term on the right hand side decays faster in n than other terms, and hence for sufficiently large n it can

 ISBN: 978-1-61208-450-3

be made smaller than a given probability α. Thus, the expected error $I(\hat{R})$ of the response mean used in the previous section is within ϵ of the optimal error $I(f^*)$ with a probability that increases with the number of observations, and is independent of the underlying distributions. An indirect evidence of this is the increased stability of the response mean as we increase the number of connection degradation events from 10 to 100 in Figures 9(a) and 10, respectively. In summary, this analysis provides a statistical basis for using the response mean $\hat{R}$ as an approximation to the underlying response regression $\bar{R}$ in comparing different methods and configurations.

V. Conclusion

We considered two sites connected over a dedicated, long-haul connection, which must fail-over to a standby connection upon degradations that affect the host-to-host application performance. Current solutions require significant customization due to the vendor-specific software of network devices and applications, which have to be repeated with upgrades and changes. Our objective is to exploit recent SDN and Network Function Virtualization (NFV) technologies to develop faster and more flexible fail-over solutions implemented entirely in software. We first presented a light-weight method that utilizes host scripts to monitor the connection rtt and OpenFlow dpctl API to implement the fail-over. We then presented a second method using two OpenDaylight (ODL) controllers and REST interfaces. We performed experiments using a testbed consisting of HP and Cisco switches connected over long-haul connections emulated in hardware. They showed that both methods restore TCP throughput, but their comparison was complicated by the restoration dynamics of TCP throughput which contained significant statistical variations. To account for them, we developed the impulse-response method to estimate the response regressions, which enabled us to compare these methods under different configurations, and conclude that on the average the dpctl method responds several seconds faster than ODL method.

It would also be of future interest to generalize the proposed methods to trigger fail-overs based on parameters of more complex client-server applications. The performance analysis of such methods will likely be much more complicated since the application dynamics may be modulated by the already complicated TCP recovery dynamics. Currently there seems to be an explosive growth in the variety of SDN controllers [4], including open source and vendor specific ones. Also, there is a wide variety of implementations of OpenFlow standards by switch vendors [1], ranging from building additional software layers on existing products to developing completely native software stacks. It would be of future interest to develop general performance analysis methods that enable us to compare various SDN solutions (that comprise of controllers, switches and application modules) for more complex scenarios such as data centers and cloud services distributed across wide-area networks. In particular, it would be of interest to develop methods that directly estimate the performance differences between different configurations from measurements using methods such as the differential regression [21].

Acknowledgment

This work is funded by the High-Performance Networking Program, Office of Advanced Computing Research, U.S. Department of Energy, and by Extreme Scale Systems Center, sponsored by U. S. Department of Defense, and performed at Oak Ridge National Laboratory managed by UT-Battelle, LLC for U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, 2015, pp. 14–76.

[2] Y. D. Lin, D. Pitt, D. Hausheer, E. Johnson, and Y. B. Lin, "Software-defined networking: Standardization for cloud computing's second wave," IEEE Computer: Guest Editorial, November 2014, pp. 19–21.

[3] E. B. Y. M. B. Al-Somaidai, "Survey of software components to emulate openflow protocol as an sdn implementation," American Journal of Software Engineering and Applications, vol. 3, no. 6, 2014, pp. 74–82.

[4] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in Proc. 2nd USENIX Conf. Hot Topics Manage.Internet Cloud Enterprise Netw. Services, 2012, p. 10.

[5] J. Tourrilhes, P. Sharma, S. Banerjee, and J. Pettit, "SDN and OpenFlow evoltion: A standards perspecitve," IEEE Computer, November 2014, pp. 22–29.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. V. aand J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined WAN," pp. 3–14, 2013.

[7] K. Phemius, M. Bouet, and J. Leguay, "DISCO: distributed multi-domain sdn controllers," 2013, http://arxiv.org/abs/1308.6138.

[8] "Opendaylight," www.opendaylight.org.

[9] V. Tiwari, R. Parekh, and V. Patel, "A survey on vulnerabilities of openflow network and its impact on sdn/openflow controller," World Academics Journal of Engineering Sceinces, vol. 1, 2014, pp. 1005:2–5.

[10] Y. Srikant, The Mathematics of Internet Congestion Control. Birkhauser, Boston, 2004.

[11] W. R. Stevens, TCP/IP Illustrated. Addison Wesley, 1994, volumes 1-3.

[12] N. S. V. Rao, J. Gao, and L. O. Chua, "On dynamics of transport protocols in wide-area internet connections," in Complex Dynamics in Communication Networks, L. Kocarev and G. Vattay, Eds. Springer-Verlag Publishers, 2005.

[13] V. N. Vapnik, Statistical Learning Theory. New York: John-Wiley and Sons, 1998.

[14] C. E. Rothenberg, R. Chua, J. Bailey, M. Winter, C. N. A. Correa, S. C. de Lucena, M. R. Salvador, and T. D. Nadeau, "When open source meets network control planes," IEEE Computer, November 2014, pp. 46–53.

[15] N. S. V. Rao, S. E. Hicks, S. W. Poole, and P. Newman, "Testbed and experiments for high-performance networking," in Tridentcom: International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2010.

[16] "Open networks foundation," www.opennetworking.org.

[17] "Software defined networks," cisco Systems, www.cisco.com.

[18] "Software defined networking," hP, www.hp.com.

[19] I. Rhee and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," in Proceedings of the Third International Workshop on Protocols for Fast Long-Distance Networks, 2005.

[20] M. Anthony and P. L. Bartlett, Neural Network Learning: Theoretical Foundations. Cambridge University Press, 1999.

[21] B. W. Settlemyer, N. S. V. Rao, S. W. Poole, S. W. Hodson, S. E. Hicks, and P. M. Newman, "Experimental analysis of 10gbps transfers over physical and emulated dedicated connections," in International Conference on Computing, Networking and Communications, 2012.

 ISBN: 978-1-61208-450-3

A Channel Utilization Method for Flow Admission Control with Maximum Network Capacity toward Loss-free Software Defined WMNs

Masaki Tagawa
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
e-mail: tagawa.masaki.td4@is.naist.jp

Yuzo Taenaka
Information Technology Center
The University of Tokyo
Tokyo, Japan
e-mail: taenaka@nc.u-tokyo.ac.jp

Kazuya Tsukamoto
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan
e-mail: tsukamoto@cse.kyutech.ac.jp

Suguru Yamaguchi
Graduate School of Information Science
Nara Institute of Science and Technology
Nara, Japan
e-mail: suguru@is.naist.jp

***Abstract*—This paper proposes an efficient multi-channel utilization method that reduces packet loss on software defined multi-channel wireless mesh network (SD-WMN). In SD-WMN, multiple channels can be used in parallel along with a route. To utilize multiple channels efficiently, we proposed a channel utilization method that balances the channel load by exploiting the flow-based control of OpenFlow. However, packet loss often occurs under this proposed method. Specifically, this method balances the network load, thereby decreasing the available capacity of each channel. Along to this, an acceptable flow also becomes smaller. A channel may run out of capacity when a large flow arrives, even if the total available capacity provided by SD-WMN is sufficient for that flow. Thus, in this paper, we propose a new method that is optimized to increase the available capacity on some of all channels by intentionally making the channel utilization imbalanced. When a new flow arrives, the flow is transmitted on the channel, thereby minimizing the possibility of packet loss. We then conduct some experiments in various scenarios and show that the method can extremely decrease the packet loss at the time when a flow arrives.**

Keywords*—*capacity management; traffic engineering; multiple channels; OpenFlow; wireless mesh network.

I. INTRODUCTION

With the popularization of smart devices, the amount of mobile data traffic from such devices is expected to increase nearly tenfold between 2014 and 2019 [1]. To offload this heavy traffic from cellular networks, access points (APs) which are base stations on a wireless local area network (WLAN) are spreading. However, APs tend to be densely placed near each other and often suffer from severe radio interference. Furthermore, the coverage area of each AP overlaps with each other, thereby resulting in small service coverage. From these reasons, the amount of offloading traffic is still limited under current WLAN environments. Therefore, it is essential to extend WLAN coverage.

Wireless mesh network (WMN) is a good solution to extend WLAN coverage. In WMN, all APs construct a multi-hop wireless backbone network (WBN) by using the same channel and only some specific APs provide the Internet connectivity to other APs through the WBN. The multi-hop network contributes to large coverage but limits the network capacity because the capacity of a single channel is theoretically limited. In order to exploit WMN for the purpose of coverage extension, we need to increase the network capacity by effectively aggregating multiple channels.

In the previous work, we proposed a novel WMN architecture that utilizes multiple channels in parallel [2]. However, the network capacity provided by a WBN strongly depends on how to utilize multiple channels. We thus proposed a channel utilization method to maximize the network capacity. The method balances the network load by switching transmission channels for each flow while considering the impact of radio interference inside the WMN. This method is called Traffic Volume Balancing Method between Interference APs (TBI) [3]. However, TBI often causes packet loss when a new flow arrives. Since TBI utilizes all channels equally, the available capacity of each channel is decreased in accordance with the increase of flows. More specifically, the acceptable flow size of every channel becomes smaller. As a result, when a flow arrives, no channel can transfer the flow without packet loss even if the available capacity provided by all channels is enough for transmitting the flow. To avoid such situation, the available capacity of some channels should be maximized as much as possible even when there are multiple flows.

In this paper, we further develop the channel utilization method to avoid packet loss at the time of flow arrival. The method tries to maximize the available capacity on some of the channels on a WBN. That is, the method fills channels with flows, thereby making the channel utilization imbalanced. The channel is selected in order from least available capacity for keeping channels with large available capacity to carry a new flow with minimum possibility in packet losses.

The rest of the paper is organized as follows. Section II is an overview of related work. In Section III, we outline an overview of our previous studies and point out potential packet loss on channel utilization methods which balance the

 ISBN: 978-1-61208-450-3

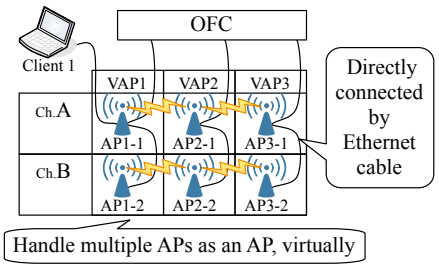


Figure 1. WBN architecture with virtual AP.

network load on multiple channels. Section IV develops the channel utilization method to avoid packet loss by making the channel utilization imbalanced. In Section IV, we implement and examine the proposed method in a real testbed and demonstrate the effectiveness of our method. Finally, we conclude this paper with conclusions and future work in Section VI.

II. Related work

To date, many studies focusing on routing protocols, channel assignment, and a MAC protocol, have been conducted for increasing network capacity [4–9]. Furthermore, many multi-channel routing protocols are conducted in conjunction with channel assignment [5–9]. In essence, these studies dynamically switch paths by updating the routing table in response to the change in the network condition. However, these protocols cannot simultaneously use multiple paths between two neighboring APs because the routing table generally contains only a single path (i.e., one channel) to reach a neighboring AP. Thus, it is necessary to propose a channel utilization method that uses all available channels between neighboring APs effectively and simultaneously.

III. Previous work

This section outlines an overview of our previous studies [2][3]. We first introduce the architecture of our WBN in Section III-A and then briefly describe TBI in Section III-C. Finally, we point out potential packet loss on TBI in Section III-C.

A. Multi-channel WBN based on OpenFlow control

Single channel WMN can suffer from serious network capacity degradation due to the nature of the wireless medium. Therefore, in order to increase the network capacity, effectively aggregating multiple channels is needed. To increase the number of channels used for the WBN, we employ a virtual AP (VAP) as shown in Figure 1. The VAP consists of multiple APs, each of which uses a single but different channel for the WBN and are directly connected by Ethernet cable. From this structure, we can flexibly increase the number of channels used for the WBN in accordance with the number of APs belonging to the VAP. Note that, to easily indicate each VAP, a VAP have its own identification number (VAPID) denoted as "VAPX". In addition, each AP is uniquely identified as "APX-Y", which combines a VAPID ($= X$) and a sequential number (APID) ($= Y$).

To flexibly use channels on the WBN, we employ OpenFlow, which enables us to use programs to control flow transfer. In this study, a flow is identified by 4-tuples (source/destination IP address and port number). An OpenFlow network consists of an OpenFlow Controller (OFC) and an OpenFlow Switch (OFS). The OFC connects with all OFSs (=APs) and determines flow control rules (flow entries) when necessary. All OFSs actually handle flows by following flow entries which are cached in a local database (flow table). By exploiting flow based control mechanism of OpenFlow, we can flexibly select a path (channel) for each flow at each hop. That is, we dynamically control channel utilization on the WBN.

B. Proposed channel utilization methods

We introduce a channel utilization method, TBI, which was proposed in the previous work[3]. The method balances the network load by controlling transmission channels of each flow. TBI treats the byte count transmitted for certain duration as the network load. Since there is radio interference between hops in a multi-hop network, TBI also considers radio interference inside the WBN. More specifically, the OFC makes all OFSs (APs) send probe packets to each other and then identifies interference APs for each AP. Then, based on this recognized interference (i.e., radio range), the OFC calculates the total amount of bytes transmitted within radio range and uses this as the network load.

TBI tries to balance the network load, however, the network load caused by new flows is unknown at first. Therefore, the OFC cannot find an appropriate channel for a flow so as to balance the network load at a flow arrival. Thus, in TBI, the OFC tentatively selects a channel to carry the flow. Once the OFC identifies the traffic volume of the flow, the OFC tries to change the channel to carry the flow to balance the network load among all channels in the WBN. TBI achieves these two allocations by two types of processing: the initial allocation and the late binding, respectively.

In the initial allocation, the OFC tentatively selects a channel with lesser network load to avoid packet loss as much as possible. The initial allocation is conducted upon the arrival of each new flow. On the other hand, the late binding process tries to make the traffic volume of each channel on a VAP balanced in case of imbalance utilization of channels. For the late binding, the OFC periodically checks the transmitted byte count of each flow. Then, in order to obtain the network load of each channel, the OFC calculates the sum of transmitted byte counts of each flow on each channel. If the OFC identifies an imbalanced channel utilization, the OFC switches the transmitting channel of some flows in the WBN.

C. Potential packet loss due to balancing mechanism

TBI balances the network load by equally transmitting flows in terms of transmitted bytes as shown in Figure 2. Along with the increase of flows, the available capacity decreases on all channels. Although TBI transmits a new flow on a channel with the lowest network load, the available capacity of a single channel may not be enough for the flow and packets

 ISBN: 978-1-61208-450-3

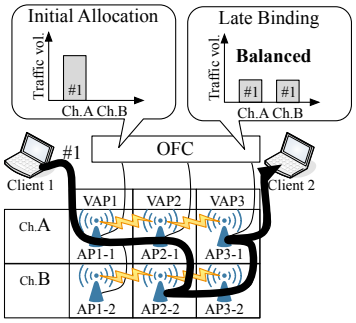


Figure 2. Example of flow allocation with TBI.

are inevitably dropped. However, even in this case, the WBN may still have sufficient available capacity for the flow with regards to the sum of available capacity in all channels. From these considerations, we conclude that the balancing channel utilization method increases the possibility of packet loss.

IV. Proposed method

We propose a new channel utilization method to minimize the number of packet losses. Since the reason for packet loss is the balancing mechanism as described in Section III-C, we intentionally create an imbalanced channel utilization. That is, the available capacity is maximized on some of the channels and other channels are filled by flows. If at least one channel can have a large available capacity, the possibility of packet loss could be minimized by transmitting a new flow on such a channel. In order to fill the channel with flows without packet loss, it is necessary to calculate the available network capacity of each channel. The calculation method of these capacity is described in Section IV-A. To achieve flow allocation according to the concept, an arriving flow is (1) initially transmitted on a tentative channel and then (2) moved onto a channel to fill the channel. Also, some flows are switched to a channel whenever a flow transmitted on the channel is ended. These processes are described in Section IV-B, Section IV-C and Section IV-D, respectively.

A. Calculation of available capacity

To maximize the available capacity in some channels, flows should be packed into the smallest number of channels. To design a method according to the concept, it is necessary to precisely know the available capacity on every channel because without this information, a channel may be saturated due to running out of capacity.

In the wireless network, it is easy to measure the amount of transmitted traffic but the available capacity is not so. In this study, we estimate the available capacity based on the information about transmitted traffic. We first obtain the statistical information on transmitted traffic (i.e., the number of transmitted packets and the amount of transmitted traffic) for every channel on every VAP. In this measurement, we collect it twice for the predetermined interval (0.5 seconds in this study) and calculate its difference. This means that the calculated packets are transmitted during the interval. We also measure the physical link rate of every channel on every hop.

We next estimate the time that was necessary to transmit the packets (i.e., channel occupancy time). It can be calculated by dividing the amount of traffic (including frame/packet headers) by the physical link rate and then adding overhead (e.g., DIFS and SIFS) of every packet transmission. Since the statistical information was measured for the predetermined interval, we can easily find the duration that was not occupied, i.e., the differences between the predetermined and the calculated channel occupancy time. Finally, we can calculate the available capacity by multiplying the unoccupied duration by the physical link rate.

B. Tentative channel allocation for arrival flow

In order to minimize the possibility of packet loss in TBI, new flows are transferred with a channel which has largest available network capacity. However, it is important to note that the available network capacity should be shared by some flows. For example, when multiple flows arrive almost simultaneously, TBI uses the same information about available network capacity to select a tentative channel to transmit each of them. As a result, they are transmitted on the same channel and share the available network capacity of a single channel. If other channels are free, these channels should be selected for some of the flows to avoid packet loss as much as possible. From this consideration, we additionally consider the expected occupation capacity on a channel for each arriving flow. That is, we take into account the number of flows, which is transmitted on a channel but its occupation capacity is not identified yet. The detailed process is described below.

The tentative allocation is conducted at each arrival of flow. The OFC first obtains the available network capacity of each channel (referred to as Ac) as described in Section IV-A. Then, the OFC collects the number of flows which are supposed to be sharing the available network capacity (this number is referred to as n). To obtain this, the OFC checks the number of flows whose statistical information (i.e., the number of transmitted packets and the amount of transmitted traffic) has not yet been identified. Note that, to calculate the expected occupation capacity, we increase the number by one as if the focusing flow is transmitted on the channel.

The OFC next estimates the expected occupation capacity of the flow on each channel by $Ac/(n+1)$. Then, the OFC identifies the channel with the largest expected value. Note that this process is performed on all hops upon which the flow will pass. Finally, the OFC allocates the flow to the identified channel by registering flow entries to OFSs.

C. Packing flows into the smallest number of channels

Since the OFC cannot obtain the statistical information of a flow at its arrival, the OFC tentatively allocates the flow to the channel. To minimize the possibility of packet loss in the tentative channel allocation, the WBN should keep some channels free. To achieve this, we pack as much flows into the smallest number of channels. Note that in accordance with the increase of flows, the number of channels which are filled by flows inevitably increases. In order to obtain largest available

 ISBN: 978-1-61208-450-3

capacity among these channels, we select a channel to be filled next in order from the least available capacity to the largest.

After the tentative channel allocation at arrival of a flow, the OFC tries to move the flow to a specific channel. For this migration, the OFC periodically collects statistical information for every flow and calculates the available capacity on each channel as described in Section IV-A. Then, once the statistical information on flows which are transmitted on a tentative channel is acquired, the flow should be migrated to a specific channel.

The OFC then tries to find a channel to migrate the flow. The channel is selected in order of the least available capacity at that time. Then, the OFC compare the available capacity of the selected channel with the amount of the flow. If the amount of the flow is smaller than the available capacity, the flow is switched onto the channel. Otherwise, the OFC selects a next candidate channel and conducts the same process until finding a channel to be used. If the flow cannot be allocated to any channel other than the current channel, the flow remains on the channel.

D. Handling flow completion

When a flow is ended, the available network capacity increases at the channel that transferred the flow (this channel is referred to as B). In this case, some other flows can be packed into channel B, thereby increasing the available capacity on other channels. Therefore, we try to move flows to channel B and fill the gap caused by the completion of the flow so that other channel could be used for a tentative channel at flow arrival. To achieve this, the flow is selected on a channel that carries flows but have the largest available capacity. If no flow is found on the channel, the OFC focus on the next channel that has the next larger available capacity. Note that, if any flow cannot fit the gap, the OFC left the gap to pack a new flow that should arrived later. The detailed procedure is described step by step below.

OpenFlow has a function in which the OFS (AP) automatically notifies the OFC about the inactiveness of flow entries which handle the packet transmission of each flow on the OFS. Therefore, we regard the flow as completed when this notification is generated. When the OFC finds the channel whose flow is completed (this channel is referred to as B), the OFC tries to change transmitting channel of flows from other channels to channel B as shown in Figure 3. Specifically, once the OFC finds the channel B, the OFC checks the available network capacity of each channel in descending order. If the available capacity of the channel is greater than that of channel B, the OFC checks flows on the selected channel.

The OFC tries to identify flows on the channel to fill channel B. To achieve this, the OFC calculate the occupancy network capacity of each flow in the same way described in Section IV-A. The OFC then compares this and the available capacity of the channel, then decides whether the flow can be sent on the channel without packet loss. At this time, in order to migrate flows as much as possible, the OFC selects target flows in descending order of the occupancy network

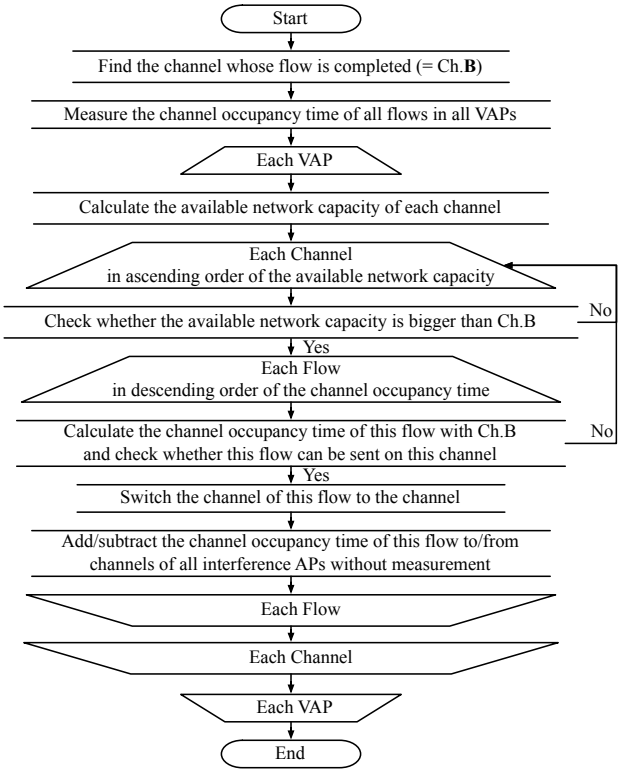


Figure 3. Flowchart of handling flow completion.

capacity. Finally, the OFC changes the transmission channel of the identified flow to channel B. In this way, the OFC keeps a specific channel to be filled by flows whenever a flow ends.

V. Performance evaluation

We evaluate the effectiveness of the proposed methods in two types of traffic. In Section V-A, we describe the experimental environment. In Section V-B, we evaluate the effectiveness with simple traffic and use the flow allocation to show how our method works. Finally, in Section V-C, we evaluate the practical effectiveness of our proposed method by complex traffic.

A. Experimental environment

The experimental topology is a 3-hop 4-channel WBN. Each AP is placed 70 cm apart as shown in Figure 4. A Buffalo WZR-HP-AG300H with OpenWrt [10] firmware was used as the AP hardware. We additionally installed the Open vSwitch [11] to the OpenWrt firmware, which allows us to operate an AP as an OFS. In the WBN, we use IEEE802.11a operating on channels 100, 112, 124 and 136. In addition, physical link rate on each channel is fixed as 54 Mbps. With this WBN, we also prepare two PCs (Client 1 and Client 2) to send/receive experimental traffic and connect them to AP1-1 and AP4-1, respectively.

We employ Trema [12] as an OFC framework on which the proposed method is implemented. Since we focus on the performance of the channel utilization method, the following

 ISBN: 978-1-61208-450-3

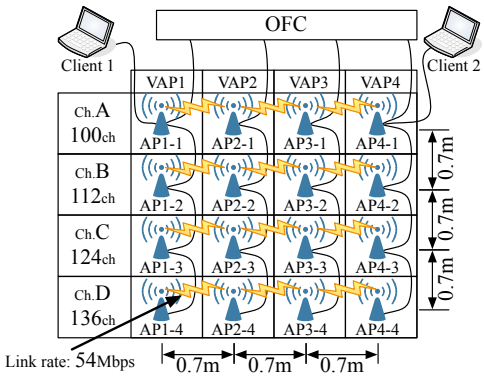


Figure 4. Network topology as in performance evaluation.

pairs of <OFS, OFC>, <Client 1, AP1-1>, and <Client 2, AP4-1> are directly connected by Ethernet cable to avoid the degradation of the communication performance inside the pair.

B. Basic performance evaluation: Simple traffic scenario

In this section, we examine the effectiveness of the proposed methods with a simple scenario. We prepare two types of WBNs, TBI-based WBN and proposed method based WBN. Note that in this experiment, we use only two channels in a testbed. In this experiment, two UDP flows of 5 Mbps and one UDP flow of 10 Mbps with 1500 byte packets ware transmitted from Client 1 to Client 2 in turns at 5 second intervals. Figure 5 demonstrates the time series variation of the packet loss. The TBI-based WBN has many packet losses after starting the third flow. In TBI, each 5 Mbps flow is transmitted on each channel at that time as shown in Figure 6(a). The network loads are actually balanced among these channels. However, this balanced utilization causes saturation on one of the channels after the third flow arrives. Moreover, the number of packet losses fluctuates drasticallly. Once packet loss occurs as a result of channel saturation, the statistical information (i.e., the number of transmitted packets and the amount of transmitted traffic) of each flow is changed to randomly smaller value than actual one. Therefore, the late binding processing attempts to balance the network load among all channels based on incorrect statistical information. The late binding is periodically conducted and thus flow transmission is frequently changed. This causes fluctuation on the number of packets which are lost.

By contrast, the proposed method based WBN does not experience any packet loss during this experiment. Two 5 Mbps flows are packed into a single channel and thus there is enough network capacity for the third flow on either channel as shown in Figure 6(b). Thus, we can say that the proposed channel utilization method can avoid packet loss at flow arrival.

C. Practical performance evaluation: Complex traffic scenario

The reason for packet loss in TBI is unpredictability of new flow arrival and its specification. In this section, we compare

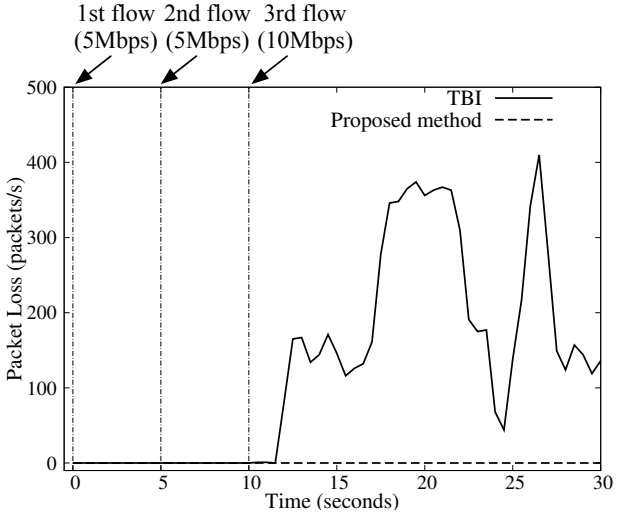
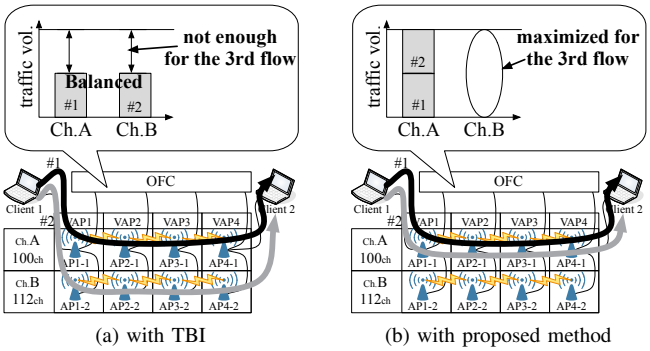


Figure 5. The time series behavior on packet loss.



(a) with TBI (b) with proposed method

Figure 6. Flow allocation after the 2nd flow arrival.

our method with TBI by random traffic scenario. We first measure the maximum network capacity by a single channel in our testbed. We transmit a flow from Client 1 to Client 2 by only one channel. The maximum data rate without packet loss on every channel was 9 Mbps. Thus, we generate random traffic for this experiment. Specifically, the traffic is generated by the following rules and the experiment is concluded after 100 flows are generated.

1) The flow arrival interval should be less than 1 second.
2) Each flow is kept for more than 1 second but not more than 10 seconds.
3) A data rate of each flow is fixed but should be less than 9 Mbps (i.e., the maximum network capacity in the single channel).
4) The total amount of traffic generated in parallel is kept less than 36 Mbps (i.e., the maximum network capacity in the WBM).

In this experiment, the generated traffic is transmitted from Client 1 to Client 2 shown in Figure 4. This experiment was conducted 9 times on the same traffic pattern. The number of packet losses was calculated by comparing the

 ISBN: 978-1-61208-450-3

TABLE I. THE NUMBER OF PACKET LOSSES WITH COMPLICATED TRAFFIC (PACKETS).

Applied Method	Maximum	Median	Minimum
TBI	6826	4229	1196
Proposed	8	7	4

transmitted/received bits captured by tcpdump on Client 1 and Client 2 during this period. Note that the number of transmitted packets was 197,906 packets.

The maximum, median, and minimum number of packet loss are listed in Table I. The traffic should be carried on this WBN without any packet loss because the maximum amount of traffic generated in parallel is less than the maximum network capacity in the WBM. Furthermore, the data rate of each flow is under the network capacity of a single channel. Therefore, the WBN have enough available capacity at any time in this experiment. However, in the case of TBI, there is a significant amount of packet loss. With late binding processing in TBI, flows are allocated in order to balance the network load between each of the channels. As we pointed out in Section V-B, the statistical information of each flow is calculated to be smaller than actual information in TBI. As a result, even if we conduct experiments with the same traffic, the flow allocation on each experiment will differ. This is why there is a big difference between the maximum number of packet loss and the minimum one on TBI. In contrast to TBI, the proposed channel utilization method successfully reduces the number of packet losses significantly in a stable manner. Even if a new flow arrives and its specifications are unpredictable, the proposed method allocates these flows to channels providing enough capacity and packed the flow into specific channels soon, thereby avoiding saturation of all channels in a WBN. Therefore, the proposed method has a potential to avoid packet loss by utilizing multiple channels efficiently.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a channel utilization method that enables the consideration of the available network capacity on each channel in a WBN. With channel utilization methods which equally utilize all channels, the available capacity of each channel is also decreased and accordingly the acceptable flow size of each channel becomes smaller. As a result, even if enough available capacity provided by all channels still remains for new flow, one channel may be saturated due to running out of the capacity once a flow arrives.

In this study, we propose a new flow allocation method that makes the channel utilization imbalanced. That is, the method maximizes the available capacity on some of the channels and fills other channels with flows. In case of flow arrival, this method allocates new flows to the channel that brings the largest expected occupation capacity for the flow. As a result, the possibility of packet loss is minimized by utilizing multiple channels efficiently. We then implemented the method based on the OpenFlow framework and conducted performance evaluations in a real testbed. In the experiment, we used complex traffic that should be carried on the WBN without any packet loss. From the results, we conclude that the proposed method can reduce the number of lost packets significantly. In the future, we plan to continue with the examination of methods for effectively utilizing multiple channels in environments having more clients (i.e., more complex topology).

ACKNOWLEDGEMENTS

This study is partly supported by The Telecommunications Advancement Foundation, Japan.

REFERENCES

[1] Cisco Visual Networking Index, Global Mobile Data Traffic Forecast Update, 2014-2019, http://www.cisco.com / c / en / us / solutions / collateral / service - provider / visual-networking-index-vni/white_paper_c11-520862.pdf [retrieved: Jan. 2016].

[2] Yuzo Taenaka and Kazuya Tsukamoto, “An efficient traffic management framework for multi-channel wireless backbone networks”, IEICE Communications Express, vol. 3, no. 3, 2014, pp. 98–103.

[3] Yuzo Taenaka and Kazuya Tsukamoto, “A radio interference aware dynamic channel utilization method on software defined WMN”, In Proceeding of the 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC 2015), Nov. 2015.

[4] A. Antony Franklin, Athula Balachandran, and C. Siva Ram Murthy, “Online reconfiguration of channel assignment in Multi-Channel Multi-Radio wireless mesh networks”, Elsevier Computer Communications, vol. 35, no. 16, Sept. 2012, pp. 2004–2013.

[5] Weisheng Si, Selvadurai Selvakennedy, and Albert Y. Zomaya, “An Overview of Channel Assignment Methods for Multi-radio Multi-channel Wireless Mesh Networks”, Journal of Parallel and Distributed Computing, vol. 70, no. 5, May 2010, pp. 505–524.

[6] Saad Mustafa et al., “Stable-path multi-channel routing with extended level channel assignment”, International Journal of Communication Systems, vol. 25, no. 7, 2012, pp. 887–902.

[7] Eiman Alotaibi and Biswanath Mukherjee, “Survey Paper: A Survey on Routing Algorithms for Wireless Ad-Hoc and Mesh Networks”, Computer Network, vol. 56, no. 2, Feb. 2012, pp. 940–965.

[8] Parth H. Pathak and Rudra Dutta, “A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks”, IEEE Communications Surveys and Tutorials, vol. 13, no. 3, Sept. 2011, pp. 396–428.

[9] Djohara Benyamina, Abdelhakim Hafid, and Michel Gendreau, “Wireless Mesh Networks Design - A Survey”, IEEE Communications Surveys and Tutorials, vol. 14, no. 2, 2012, pp. 299–310.

[10] OpenWrt, https://openwrt.org/ [retrieved: Jan. 2016].

[11] Open vSwitch, http://openvswitch.org/ [retrieved: Jan. 2016].

[12] Trema, http://trema.github.io/ [retrieved: Jan. 2016].

 ISBN: 978-1-61208-450-3

Implementation of Virtualization in Software Defined Networking (SDN) for Data Center Networks

Nader F. Mir, Jayashree N. Kotte, and Gokul A. Pokuri
nader.mir@sjsu.edu
Department of Electrical Engineering
San Jose State University
San Jose, CA, 95195

Abstract—**Software Defined Networking (SDN) is an emerging technology in IT industry. There is enormous pressure on network operators to change the conventional network architecture to accommodate the rising need for innovative services and fluctuating demands. The reasons behind the network architecture evolution are heavy traffic, scalability and enormous bandwidth shortage. With the implementation of SDN in a Data Center Network (DCN), providing centralized control can eliminate major issues of routing. In this paper, we develop a user application to scale the DCNs and analyze SDN's performance. We consider networks connecting 8 hosts to 512 hosts and measure performance metrics in terms of latency, packet drop rate and bandwidth. The tools being used are Mininet, vSwitch, OpenFlow controller and VMware workstation. Python language is used to bind the tools together. Customized topologies are created and implemented using OpenFlow controller to determine SDN's efficiency.**

Keywords-SDN; software defined networking; virtualization; data centers.

I. INTRODUCTION

During early 2004, networks had distributed configurations. After 2004, the centralization of network control came into existence with the emergence of Border Gateway Protocol (BGP) that run on Routing Control Platforms (RCP). RCP was generalized for decision planes, which was responsible for computing routes. Also, RCP was generalized for data plane that forwards packets [1]. In 2008, OpenFlow was presented, whose fundamental roots came from RCP. The idea behind OpenFlow was the decoupling of the control and data planes.

Virtualization in networks was another effective trend that allows network administrators to run applications on fewer physical servers [2][3]. No business can afford application downtime and virtualization provided a way to decrease application downtime. High availability and fault tolerance are built right into the platform. Virtualization provides bigger savings by letting a network administrator run many apps on fewer servers at the industry's lowest net virtualization cost.

In general, any routing device contains two planes of operations: control plane and data plane. The task of the data plane is to forward packets to destinations. Routers determine the path for routing packets based on their respective routing tables. The control plane computes these routing tables. The ideas of network virtualization and decoupling data and control planes resulted in the *Software Defined Networking* (SDN).

SDN enables application of virtualization principles on network infrastructure by abstracting network resources, pooling and automating them to outshine the limitations of the network architectures [4]. SDN changes the device-level configuration by centralizing the control plane [5][6].

In SDN, the control lies in the centralized controller, which runs on top of the data plane. This controller provides a complete picture of the network and is responsible for controlling all the routers in the network. This feature of the SDN controller gives network operators network-wide control. The separated control plane offers faster innovation as the orchestration of the network constituents is done on an open interface.

OpenFlow is an open interface standard that enables network operators to control the chips placed in a switch using software. This paved the way for better innovation and easier migration of resources with ambiguous demands from enterprises. OpenFlow brought the open vSwitch into existence [6]. Open vSwitch became the root of system architecture decoupling. It also enabled protocols under the experimenting state to co-exist with the legacy protocols in the network.

To implement SDN in any organization, the working protocol is typically OpenFlow. Communication between the centralized controller and switches is facilitated by the OpenFlow protocol [7][8]. The OpenFlow is a set of commands that the SDN controllers use to pass on the routing decisions to vSwitches for routing table updating.

A *data center* is either a physical or virtual storehouse of data. Its functions include collection, storage, management and propagation of data. A data center contains all the logs of the company varying in the level of significance. Data centers are also very similar to an operations center focused on networks. They hold many automated computers that monitor the Web activity, server access and performance of the networks.

The rest of this paper presents the components of the simulation implementation including the Mininet setup and the simulation architecture. The paper then proceeds with the implementation details, followed by some results of the simulation, including several charts presenting network latency and packet drops.

 ISBN: 978-1-61208-450-3

II. COMPONENTS OF IMPLEMENTATION

A. SDN Implementation

The infrastructure of the implementation for this article is made up of decoupled control and data planes, as mentioned earlier. The control plane is typically a software program. These software programs are usually written in high-level languages such as Python and C. The data plane is programmable hardware, which differs from traditional networks in which the data plane is not programmable.

The routing path computations made by SDN controller affect the routing table in a vSwitch. The computing decisions are propagated to the vSwitches using a set of control commands. Openflow is the set of control commands that are made use of by both data and control planes for communication.

The control planes consist of logic that controls the packet forwarding behavior of the routers and switches. It also contains configuration procedure for middle boxes, such as firewalls, and load balancers.

The data plane is responsible for forwarding packets in the network, so it contains routing tables and hardware pertaining to the network. To summarize the functionality of SDN, it is safe to say that computing the shortest paths are functions of the control plane and data plane functions handle packets and routing them from input port to output port. The separation of control and data planes facilitates the evolution of software and hardware independently. Also, it enables the controller to be programmed by high-level programs. This kind of control enforcement makes it easy to debug the network.

B. DCN Implementation

Any data center typically harbors numerous hosts. Data center networks are broadly classified into three types:

i. Server-routed networking
ii. Switch-routed networking
iii. Server-Routed Networking

A DCN, such as a server-routed networking DCN, is typically a three-tier topology. The first layer is usually an access layer that connects the server racks called top-of-rack (TOR) switches, as shown in Fig. 1. The next layer of devices consists of aggregate switches that connect the access layer and a core layer. Switching in a DCN basically involves routing a packet from a source host to a destination server. In reality, the structure of DCN has the core layer switches connected to the TOR switches, which is in turn connected a server rack. Packets within the DCN network find paths traversing through TOR and core switches but packets originating from outside the DCN are routed to the destination only if core switches provide access. Such packets from outside then find their destination server by traversing the TOR switches and server racks. The core layer switches in a data center network enforce network security and load balancing.

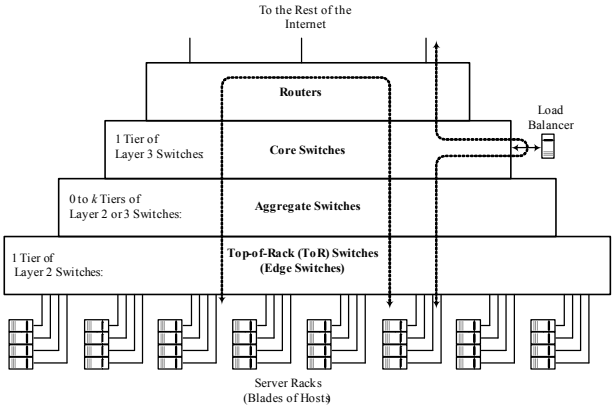


Figure 1. Data Center Architecture

C. Virtualization

Virtualization provides a layer on hardware that can be used to install an instance of an operating system. Normally, a *hypervisor* is installed on the hardware to carry out the functionality of virtualization. Hypervisors are classified into two types. The Type-1 hypervisor is installed directly on the hardware and the instances of different operating systems are installed on it. With the use of management software, it can move the instances of operating systems between physical servers based on the needs. A Type-2 hypervisor is also called hosted hypervisor. Hosted hypervisors are installed on the operating system.

D. MININET

Mininet [8] is a virtual network emulation software that is used to introduce or launch a network consisting of switches, hosts and a SDN controller. Mininet uses OpenFlow switches and routers. An example of Mininet topology used in data centers is the tree architecture, as shown in Figure 2. Network packets are routed through the SDN controller, and an action is taken for that particular packet. There is an extra latency on the first packet because of the communication of the controller. After the first ICMP request and reply, the flows are cached by the switch for a limited time. The following Mininet command creates a network with four hosts (servers) and three switches in a Tree topology: `$sudo mn --topo=tree,2,2`. We use this topology to simulate a data center network.

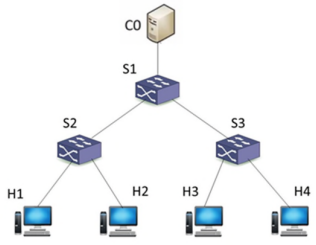


Figure 2. Tree topology, as a simple data center network

 ISBN: 978-1-61208-450-3

E. POX Controller

POX [9] is a strictly reactive cross platform controller built on Python platform with fairly simple source code. The actual modelling of POX is to dynamically respond to links and switches as their status changes to make sure those connections are always maintained. The event handler handles the status change events. It is very useful for research and experimentation. There are numerous ways to run POX and it has many components. POX controller is bundled along with Mininet.

F. OVS Switch

Open vSwitch is a virtual switch, which is used to connect hosts, in this case Virtual machines. It supports many traditional switch features like VLAN tagging, 8012.1q trunking, Spanning tree protocol, port mirroring, monitoring, tunneling (GRE, IPSec) and QoS control. Open vSwitch works in two types of modes.

1. Normal mode
2. Flow mode.

In normal mode, the switch acts like a traditional switch i.e., it acquires information about the network and computes the path. It learns the path by source MAC address and at the beginning it broadcasts and multicasts traffic until it learns all the paths. In flow mode the SDN controller configures the paths.

There are two criteria called match and action. If the condition matches the specification the respective action is executed. The match field can be layer 2, layer 3 or layer 4, therefore it can match the IP addresses, MAC addresses and transport protocols for the match condition. Open vSwitch follows a top to bottom matching approach.

Once the matching is done, the corresponding action is executed. Every flow has a specific priority assigned to it. Packet priorities are checked at the firewall level, if rejected by the switch they are rerouted to the SDN controller, which decides the entry of the packet into the network.

G. Integration of SDN in DCN

The main problem in managing a data center is supplying and migration of resources/services in response to the traffic load. SDN solves this issue by logically controlling the routing tables in the switches. If two VM's are communicating with each other, then the switches are aware of the routing paths to each of the VMs correctly. Apart from this, migration of VM becomes easy as the SDN controller regularly updates the routing table in a switch making use of a centralized database. Integration of SDN with DCN creates improved load balancing of traffic in a DCN, improving bandwidth utilization, and scaling the network with changing demands and applications. In this paper, we aim to address the scalability issue and show the efficiency of SDN in reducing the bandwidth utilization thereby improving the load balancing of traffic in a DCN. Careful study of performance parameters such as bandwidth, packet drop rate, latency performance statistics in terms of graphical representation were done.

Figure 3 shows the operation of software defined networking technology in DCN with separated control and data planes. The control plane contains the SDN controller, which is responsible for smooth flow of operations in the network. The data plane, on the other hand, merely acts as a packet forwarding backplane, which is composed of network components such as open vSwitches. Communication between the data plane and control plane takes place through the OpenFlow protocol [7]. OpenFlow protocol communicates all the routing decisions and routing table entries for routing packets to open vSwitches from SDN controller.

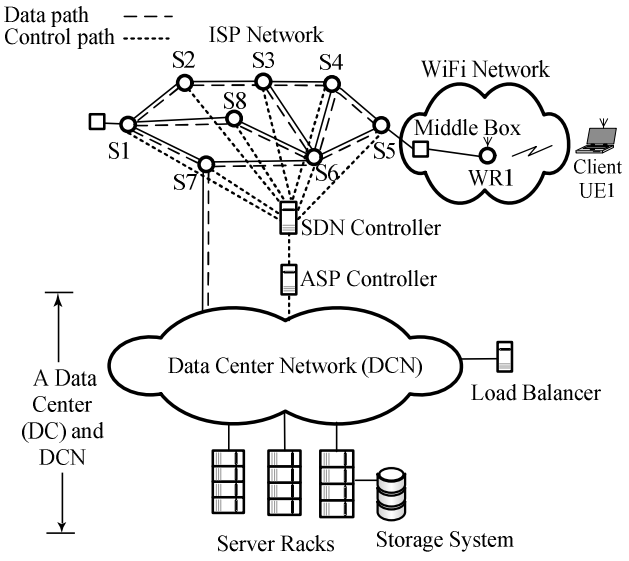


Figure 3. SDN in DCN

H. NFV

Network Function Virtualization (NFV) [4] has emerged as a result of exceeding demand of scaling of applications and services. NFV makes use of SDN to provide services in a wide spread environment.

Previously, network designers encountered a problem of last hop determination for L2 when a packet arrived. This problem was dealt with by creating the vSwitch. vSwitches addressed the problem by creating L2 VLANs. At later stages, with the increase of virtual machines, the need for L3-7 services between VMs arose. This issue gave birth to a horde of virtual versions of hardware products like vRouter, vFirewall, vNIC and virtual load balancers [4], which helped in controlling and managing the traffic in the network. Since a large number of virtual devices were being used, the need for automation/orchestration environment emerged. Due to fluctuating demands, the focus shifted to open environments for orchestration. All this complexity in management needed

 ISBN: 978-1-61208-450-3

a simplification. This was brought forth by SDN controllers such as OpenDaylight [4], which were used to simplify management of the network. To resolve the issue of scaling the server based networking across a range of servers, Network Function Virtualization was proposed.

The vendors have been demanding the service providers to change the architecture as they faced issues of the service being too slow, expensive and not being able to bring up new services. Vendors demanded very flexible software, which could be attained with NFV aided by SDN.

III. IMPLEMENTATION

VMware workstation, such as VMware Fusion or Virtualbox, was installed. Then Mininet, which is a network emulator, was installed and run on the VMware fusion. A complex virtual DCN was programmed using Python scripts in Mininet. A complex tree topology of server-routed networking scheme was used in this paper. Tests were performed on the DCN that allowed us to find the differences of traditional/conventional routing and routing using SDN.

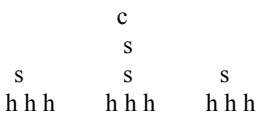
Utilizing Mininet and its Python API, an application script called Tree_1024.py was created to test the scalability of Tree_1024.py. This user application was created to reach the aim of this paper i.e., model how SDN scales in terms of latency and bandwidth as the numbers of hosts on the network increase.

Utilizing the Python API provided by Mininet makes it possible to automate the capturing of the various measurements required to reach the goals described in this paper.

Command: Function Run_mininet runs the following functions in loop for each n number of hosts (i.e., 8, 16, 32, 64, 128, 256, 512) Contains the following Mininet API function calls, TreeNet(depth=i, fanout=2, switch=OVSKernelSwitch),

Command: Treenet(depth, fanout, switch) indicates the number of levels in the network and fan-out indicates the number of leaves per switch in the network.

Eg Treenet(depth = 2, fanout=3, switch=OVSKernelSwitch) would create a network that would look like:



There are two types of switches that are used in Mininet: 1) UserSwitch – this virtual switch is created and used from user space and has no connection to the kernel; and 2) OVSwitchKernel – based on Open vSwitch but running in the Linux kernel. It handles traffic between different virtual machines on the same physical device and network, which the device is connected to.

The main program used in this paper is Tree_1024.py. This program was aimed at comparing the efficiency of SDN with increasing number of hosts. In other words, we addressed the scalability of a SDN run data center network and analyzed it through important performance metrics.

Command: Network.iperf(fmt='g')

To calculate the bandwidth between hosts, the network iperf command is utilized. This command returns the bandwidth between two hosts as a list. Here, iperf is run between the first host and the last host (e.g. in a network with 64 hosts iperf is run between host1 and host64). Using the argument fmt='g' the results are returned in Gbps (Giga bits per second).

Internally the command run as:

Local_host$iperf –p 5001 –t 5 –f g –c dest_host_ip

The iperf is run against the dest_host_ip from the local_host machine.

In the above command

-p 5001: indicates the port on which the iperf command is supposed to run.

-t 5: Indicates that iperf has to run for 5 seconds

-f g: Indicates that the bandwidth values should be returned in Gbps.

-c dest_host_ip: Indicates the destination IP address the iperf should run against.

Proc_ping

The result of the network.pingAllFull() is a Python list of lists. For each ping command run, the following list is created.

pingresultslist = [Node, dest, [sent, rec, rttmin, rttavg, rttmax, rttstd]]

Node – the IP address of the host sending the ping command.

Dest - IP address of the destination of the ping command.

Sent - Number of ICMP ping packets sent.

Rec - Number of ICMP ping packets received.

rttmin – minimum round trip time of all pings for that instance of ping command.

rttavg - Average round trip time of all pings for that instance of ping command.

rttmax – maximum round trip time of all pings for that instance of ping command.

rttstd - standard deviation of all pings for that instance of ping command.

These values are saved in variables and ping result values are stored in a csv file.

Proc_iperf

The result of network.iperf() is a list containing two values of the bandwidth for each path (Local host to remote host and vice versa).The values are cleaned (removes the Gbps in each result), averaged and stored in a variable. The values are stored in a csv file at the end.

Once the run_mininet command is run, the mini_plot function is run to create plots for the Bandwidth and Latency for the different numbers of hosts tested (8, 16, 32, 64, 128, 256, 512). All the rttavg values for each 'n' number of hosts are averaged and saved in a variable. Matplotlib that is python plotting library is used to plot the values.

The above program runs a loop to generate tree architectures by varying the depth. Depending on the depth

 ISBN: 978-1-61208-450-3

value given, it will generate tree architecture of 16, 32, 64, 256, 512 and 1024 hosts. The program also generates authentic graphs in Mininet by comparing the latency and bandwidth with the number of hosts. These graphs show the performance of SDN with increasing number of hosts.

IV. RESULTS AND OBSERVATIONS

The performance metrics that we have generated in Mininet using the automated Python script are latency and bandwidth. The values of latency and bandwidth for hosts starting from 8, 16, 32, 64, 128, 256 and 512 have been calibrated and plotted.

Fig. 4 shows bandwidth utilization in a network with SDN control when the number of hosts increases. Fig. 5 shows the latency in the same network.

In Fig. 6, the comparison of dropped packets in different network sizes using OVS and POX controller is illustrated. With the increase in network size, the number of packets dropped increases exponentially. For the networks that are created, ping time varies with respect to the controllers.

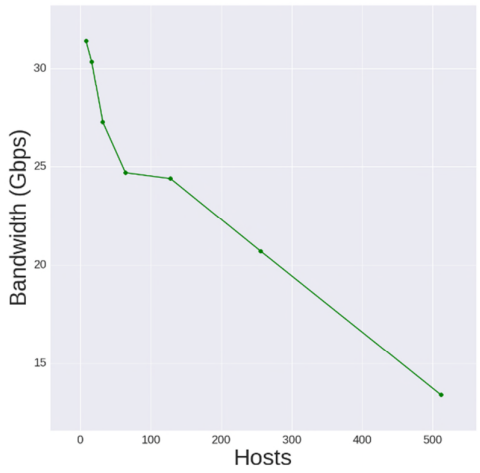


Figure 4. Bandwidth v number of hosts

Fig. 7 shows a comparison of ping time between H1 and H2 for different networks using OVS and POX controller

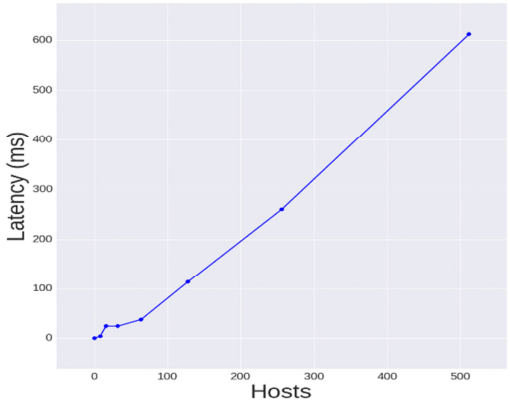


Figure 5. Latency v number of hosts

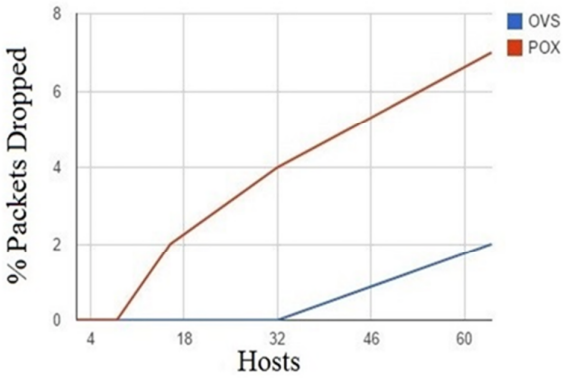


Figure 6. Percentage of packet drop vs hosts

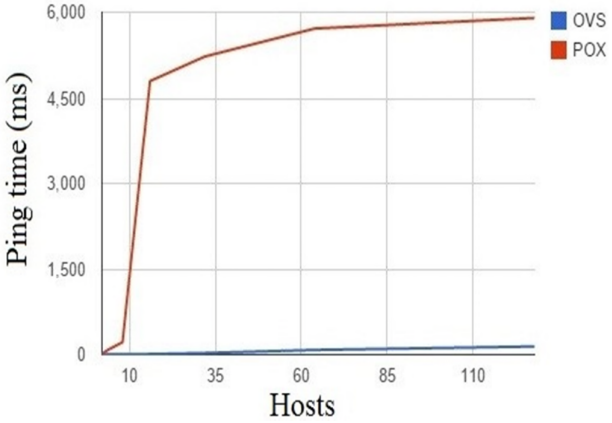


Figure 7. Ping time v/s hosts

Fig. 8 shows a comparison of ping time between a first host and a last host for different networks using OVS and POX controller. The ping results are examined between the first host and the second host of the network. With respect to POX controller design, the effectiveness to connect the source and destination host decreases.

 ISBN: 978-1-61208-450-3

The screenshots shown in Fig. 9 and 10 give the estimation of the ping time analysis between hosts. The first ping time takes more time as the route computation takes place whereas the remaining pings are faster as the routes have already been computed and cached.

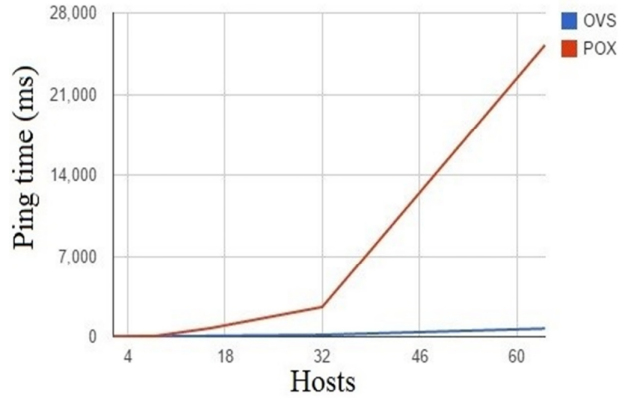


Figure 8. Ping time v/s hosts

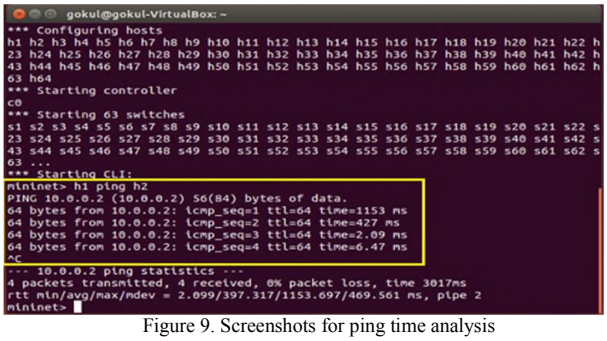


Figure 9. Screenshots for ping time analysis

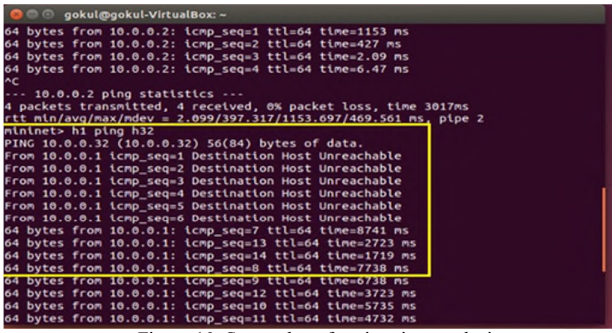


Figure 10. Screenshots for ping time analysis

V. CONCLUSION

We have implemented SDN in a data center network making use of virtualization principles. We have looked at how the network scales from 8 hosts to 512 hosts in terms of latency, packet drop rate and bandwidth. From the results, it can be safely concluded that, as the number of hosts increases, the bandwidth available decreases, and the latency between the host and packet drop rate increases accordingly. These results are in accordance with expected results at the beginning of the paper and show the efficiency of SDN over traditional network infrastructure.

VI. REFERENCES

[1] Udacity.com, 'Computer Networking Basics Training Course Online', 2015. [Online]. Available: https://www.udacity.com/course/computer-networking--ud436. [Accessed: 10- May- 2015].

[2] Vmware.com, 'Virtualization Basics, What is Virtualization: VMware | United States', 2015. [Online]. Available: http://www.vmware.com/virtualization/virtualization-basics/what-is-virtualization. [Accessed: 10- May- 2015].

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, 'OpenFLow: Enabling Innovation in Campus Networks', 2015.

[4] N. Mir, *Computer and communication networks*, 2nd ed. Upper Saddle River, NJ: Pearson Hall, 2015.

[5] T. Nadeau and K. Gray, *SDN*. Sebastopol, CA: O'Reilly Media, 2013.

[6] Sdnhub.org, 'OpenFlow version 1.3 tutorial | SDN Hub', 2015. [Online]. Available: http://sdnhub.org/tutorials/openflow-1-3/. [Accessed: 10- May- 2015].

[7] V. Tiwari, 'SDN and OpenFlow for Beginners', 2013.

[8] Minimiet, http://mininet.org. [Accessed: 10- May- 2015].

[9] POX Controller, https://openflow.stanford.edu/display/ONL/POX [Accessed: 10- May- 2015].

 ISBN: 978-1-61208-450-3

QoS Constrained Path Optimization Algorithm in NFV/SDN Environment

Yujeong Oh, Jonghun Kim, Jaiyong Lee
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
E-mail: rmang@yonsei.ac.kr, roro7773@yonsei.ac.kr, jyl@yonsei.ac.kr

***Abstract*— Network Functions Virtualisation (NFV) and Software-Defined Networking (SDN) have emerged as promising technologies in the telecommunication business. An important problem of NFV architecture is how to allocate virtualised resources to network services. Efficient resource allocation should consider not only the status of the Virtualized Network Function (VNF) but also the network condition. We propose a Quality of Service (QoS) constrained path optimization algorithm to set up the optimized VNF Forwarding Graphs (VNFFG) considering an available bandwidth. When the link bandwidth is not sufficient, competitor selection and priority switching of the super problem could find the optimized VNFFGs to accept more service requests by utilizing replaceable VNFs.**

Keywords-network functions virtualisation; software-defined networking; QoS; link capacity allocation.

I. INTRODUCTION

Network Functions Virtualisation (NFV) and Software-Defined Networking (SDN) are significant industry trends these days. The network abstraction and programmability of SDN complements the NFV architecture. ETSI NFV Industry Specification Group (ISG) proposes a NFV/SDN architecture and discusses the technical challenges [1][2]. Especially, a resource allocation to multiple network services and performance maintenance regardless of Virtual Network Function (VNF) status variation and network condition are the key challenges of NFV realization. To solve these problems, the only way is a proper link capacity allocation and VNF selection.

Some systems on the Internet provide QoS with certain packet processing in routers. There are two system models, which are, the integrated service model and the differentiated service model. For QoS service in NFV/SDN architecture, the integrated service model (e.g., RSVP-TE in MPLS) is more appropriate to provide a certain level of QoS requirement. There are many QoS parameters such as delay, bandwidth, jitter, loss probability and cost, but the crucial one is bandwidth [3][4][5]. If bandwidth for a packet flow is not enough, congestion will occur in bottleneck links, which causes severe packet drops and increases end-to-end delay. Thus, NFV/SDN architecture should manage link capacity allocation of the entire service and VNFFG simultaneously.

The paper is structured as follows: Section II outlines our proposed QoS constrained path optimization algorithm and architecture. Section III describes the competitor selection and priority switching with a simple scenario. Finally, we conclude the paper with potential future work in Section IV.

II. QOS CONSTRAINED PATH OPTIMIZATION

In the NFV/SDN environment, some VNFs can be deployed in multiple data centers, or some busy VNFs can be established in the same data center for a load sharing purpose. Those VNFs which are performing identical functions are called a VNF set and are located in multiple service provider's data centers. For instance, Figure 1 describes a NFV/SDN network of a mobile cellular network provider.

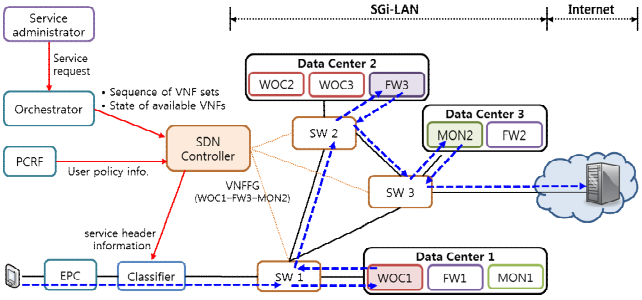


Figure 1. NFV/SDN based telecommunication network.

The telecommunication network is composed of Evolved Packet Core (EPC) and data centers, which are operated by the mobile cellular network provider. The operator manages the network between data centers with SDN. First, the service provider notifies the service request that characterizes their new value-added network service to the orchestrator. The orchestrator not only manages virtualized resources (e.g., CPU, memory, storage, switch of data centers), VNFs and lifecycle of network services, but also creates a sequence of VNF sets (e.g., WAN Optimization Controller (WOC) – Firewall (FW) – Monitoring (MON)) based on descriptors [2]. The sequence of VNF sets and its VNFs status are delivered to the VNFFG optimization algorithm on SDN controller. The SDN controller should maintain link capacity information of other network services as well as available network bandwidth information. In SGi-LAN, the telecommunication service provider operates three geographically split data centers for reasons related to operational cost or security issues. Maybe an external cloud service provider contracts with the telecommunication service provider. The VNFFG optimization algorithm creates and sets up the optimized VNFFG (e.g., WOC1 – FW3 –

 ISBN: 978-1-61208-450-3

MON2) considering VNF status, network topology and network condition. Finally, the packet flows from the user device and is forwarded to the EPC and classifier. The classifier marks network service header to packets based on pre-defined service header information that notifies what VNFFGs would be proper to this flow. Then, the SDN switch forwards the flow with pre-defined forwarding tables.

The QoS constrained path optimization algorithm of the SDN controller is composed of a super problem and sub problems, as depicted in Figure 2.

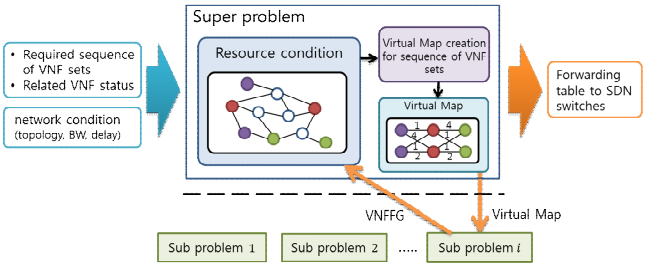
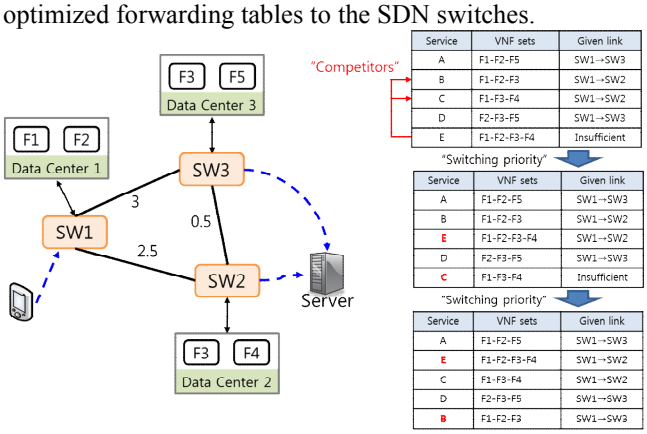


Figure 2. QoS constrained path optimization algorithm.

The super problem provides the virtual map to sub problems. The virtual map is an abstraction of the network that describes related VNFs and available bandwidth between the VNFs. The sub problems are created when a new service request arrives. Each sub problem chooses the virtual path that has the largest available bandwidth and the appropriate VNFs. The VNFFG, the result of the sub problem, is delivered taking into account the super problem's resource condition, and then the super problem deducts the resources that the VNFFG uses. When all of the sub problems are satisfied, the super problem forwards the optimized forwarding tables to the SDN switches.



Service	VNF sets	Given link
A	F1-F2-F5	SW1→SW3
B	F1-F2-F3	SW1→SW2
C	F1-F3-F4	SW1→SW2
D	F2-F3-F5	SW1→SW3
E	F1-F2-F3-F4	Insufficient

Service	VNF sets	Given link
A	F1-F2-F5	SW1→SW3
B	F1-F2-F3	SW1→SW2
E	F1-F2-F3-F4	SW1→SW2
D	F2-F3-F5	SW1→SW3
C	F1-F3-F4	Insufficient

Service	VNF sets	Given link
A	F1-F2-F5	SW1→SW3
E	F1-F2-F3-F4	SW1→SW2
C	F1-F3-F4	SW1→SW2
D	F2-F3-F5	SW1→SW3
B	F1-F2-F3	SW1→SW3

Figure 3. Competitor selection and priority switching.

III. Competitor Selection and Priority Switching

The super problem is a heuristic optimization algorithm based on competitor selection and switching priority. When the resource condition is not enough, the super problem is not able to create the virtual map for a new service request. However, in NFV/SDN environment, there is a chance that the prior sub problems could modify their VNFs and virtual path. For instance, the left side of the Figure 3 shows two different paths for using VNF 3.

The number on the links means the number of acceptable VNFFGs. The first chart of Figure 3 shows five service requests that are requested to SDN controller and their sequence of VNF sets. However, Service E is not accepted because of insufficient link resource. If prior services change their VNFFG, service E could be accepted.

To maximize network service acceptance, competitor selection of the super problem finds the prior sub problems that use bottleneck link of the unaccepted sub problem. Then, priority switching of the super problem switches the priorities of unaccepted sub problem and the sub problem, which has the lowest priority among the competitors. The second chart shows the priorities of service C and service E are changed but the service C is not accepted. Finally, priority switching of service E and the other competitor, which is service B, makes all services to be accepted. This heuristic optimization maximizes the number of service requests by utilizing replaceable VNFs.

IV. Conclusion and Future Work

In the NFV/SDN environment, a link capacity allocation is an important problem to provide a stable QoS of network services. We propose a QoS constrained path optimization algorithm considering VNF status, network topology and network condition. The super problem of VNFFG optimization algorithm manages resources and creates the virtual map about the related VNFs and virtual path between them. The sub problem chooses sufficient VNFs and links at the given virtual map. Furthermore, we propose a heuristic super problem algorithm, which uses competitor selection and priority switching to maximize the network service acceptance. For the next step of research, we will include real-world performance comparison with other researches and analytic tools of a sequence of VNF sets and network conditions to prove the resource efficiency of the proposed algorithm.

Acknowledgment

This research was supported by LG Electronics Co. Ltd.

References

[1] ETSI. Group specification. Network Functions Virtualisation; Infrastructure Overview. [online]. Available from: http://www.etsi.org/deliver/etsi_gs/ 2016.1.11

[2] ETSI. Group specification. Network Functions Virtualisation; Management and Orchestration. [online]. Available from: http://www.etsi.org/deliver/etsi_gs/ 2016.1.11

[3] Y. Wang and Z. Wang, "Explicit routing algorithm for Internet traffic engineering," Eight International Conference on Computer Communications and Networks, 1999, pp. 582–588, doi:10.1109/ICCCN.1999.805577.

[4] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," IEEE/ACM Transactions on Networking, vol. 3, no. 4, Aug. 1995, pp. 365-386, doi:10.1109/90.413212.

[5] R. Trivisonno et al., "Virtual link mapping in future SDN-enabled networks," IEEE SDN for Future Networks and Services (SDN4FNS), Nov. 2013, pp. 1-5, doi: 10.1109/SDN4FNS.2013.6702562.

 ISBN: 978-1-61208-450-3

Centralized Retransmission Management with SDN in Multihop Wireless Access Network

Bong-Hwan Oh, Jaiyong Lee
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
{crusader27, jyl}@yonsei.ac.kr

***Abstract*—In this paper, a centralized retransmission management with Software Defined Network (SDN) is proposed in order to handle packet loss in multi-hop wireless access networks. The proposed scheme manages retransmission persistence for Automatic Repeat Request (ARQ) in all wireless links, which satisfies a delay constraint and maintains end-to-end throughput. Simulation results show that the proposed scheme allows both throughput improvement and less average delay than a delay constraint by adjusting the trade-off between throughput and wireless delay.**

Keywords – SDN; retransmission; management

I. INTRODUCTION

Quality of Service (QoS) in wireless networks is an important issue because of the increase in the number of applications for delay sensitive services. Because a high wireless error and variable delay can occur in wireless links, a reliable data transmission and satisfying end-to-end delay constraint should be considered in order to support QoS in wireless networks.

Automatic repeat request (ARQ) is one of the general transport protocols in wireless networks and ARQ-based error control is an efficient methodology because ARQ can optimize packet loss recovery for a specific wireless link and can prevent a reduction of congestion window of higher layer by shielding the wireless error. However, one of the weak points of ARQ is an additional delay caused by ARQ, which can cause a degradation of high layer performance due to an inaccurate Round Trip Time (RTT) estimation. The main factor for this phenomenon is a retransmission persistence of ARQ which refers to a degree that an ARQ tries the retransmission.

There are several works attempting to improve end-to-end performance by managing the retransmission persistence for ARQ [1]-[4]. Retransmission persistence is defined as the maximum retransmission number in data link layer. In the previous works, the fundamental assumption in order to improve end-to-end performance is that wireless access networks should be single-hop wireless networks. If the network environment expands to multi-hop wireless network, the previous methods are insufficient to ensure end-to-end performance. Thus, multiple ARQs are mutually operated in order to support end-to-end performance in multi-hop wireless network.

In order to consider end-to-end performance in multi-hop wireless network, all wireless links should be estimated and handled. One solution is to apply a centralized method such as Software Defined network (SDN) [5] in order to handle all wireless links.

In this paper, we focus on an SDN-based solution for a retransmission management in multi-hop wireless network. In the proposed scheme, a SDN controller monitors all wireless links and determines the retransmission persistence of ARQ in each wireless links, which handles not only wireless delay but also end-to-end throughput.

II. CENTRALIZED RETRANSMISSION MANAGEMENT WITH SDN

The main feature of the proposed scheme is to manage the retransmission persistence for ARQ according to end-to-end throughput and wireless delay in a multi-hop wireless network. The proposed scheme is composed of two stages. In the first stage, an SDN controller monitors the state of all wireless links and compares the link state between wireless links. Then, in the second stage, centralized retransmission management is performed not only for satisfying a delay constraint, but also for improving wireless throughput depending on the state of all wireless links.

A. Monitoring status of wireless links

The main objective of the monitoring method is to discover a bottleneck link or a good link for satisfying end-to-end performance. The SDN controller monitors a link delay (D_l), a link delay variance and the retransmission persistence (RP) of ARQ in each wireless links. This link information is written in each wireless access points and the SDN controller gathers the information in all wireless access points. In the monitoring method, queuing delay is only considered in order to calculate the link delay because the link delay is mainly determined by queuing delay and other factors such as link propagation and processing delay are negligible and uncontrollable.

B. Centralized retransmission persisence management

The main object of this management is to satisfy a delay constraint as well as to maximally maintain end-to-end throughput using the retransmission persistence management. In order to perform the above operation, SDN controller should maximally increase RP in all wireless links without a delay constraint violation. It is because the increasing RP can

 ISBN: 978-1-61208-450-3

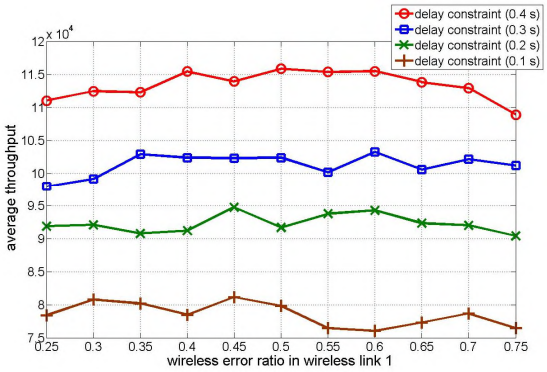


Figure 2. Throughput versus wireless link error in link 1.

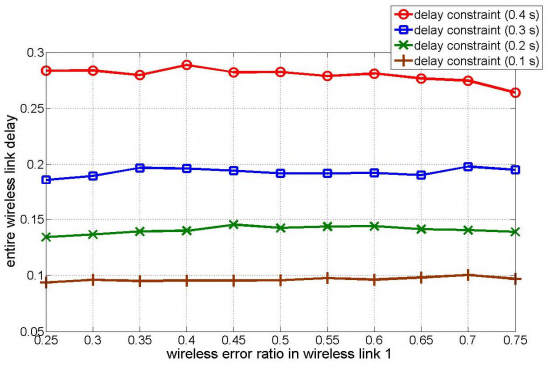


Figure 1. Entire wireless link delay versus wireless link error in link 1.

shield wireless error, which improves end-to-end throughput. Thus, the proposed management reduces or increases RP only when a cumulative delay in all wireless links exceeds a delay constraint or the cumulative delay is sufficiently small, respectively. By using the monitoring information in all wireless links, the SDN controller estimates a cumulative delay of all wireless links and discovers a bottleneck link among all wireless links. If the cumulative delay exceeds the delay constraint, SDN reduces RP in the bottleneck link. Conversely, the SDN controller increases RP in the bottleneck link when the cumulative delay is sufficiently less than the delay constraint. The delay threshold for increasing RP can be set according to the management policy.

III. SIMULATION RESULTS

A simulation is conducted in order to evaluate the performance of the proposed scheme. The client and the server are connected by two serial wireless links (wireless link1 and wireless link 2) and the wireless links are connected with the SDN controller. As a transport protocol, Transmission Control Protocol (TCP) is applied. We operate the simulation by changing the wireless link error rate. Each wireless link error rate is variable but the cumulative wireless error rate is fixed (0.2). In this simulation, the delay threshold for increasing RP is a half of a delay constraint.

Figure 1 shows the average end-to-end throughput versus error rate in wireless link 1. The x-axis indicates the error ratio of wireless link 1 to the cumulative error rate. The results show that average throughput performance tends to decrease with the decrease in the delay constraint. It is because the wireless error is less recovered in order to satisfy the delay performance. In other words, the proposed management can improve the average throughput as the delay constraint increases. It means that the proposed management can adjust the throughput performance according to the delay constraint.

Figure 2 shows the total average delay in wireless links versus the error rate in wireless link 1. The results show that the proposed management can achieve lower average delay than the delay constraint. It is because, when the delay over wireless links exceeds the delay constraint, the proposed method reduces RP in the bottleneck link, which can improve the delay performance by reducing the recovery time of ARQ due to wireless error.

IV. CONCLUSION

In this paper, a centralized retransmission management with SDN is proposed to handle wireless error in a multi-hop wireless access network. The proposed scheme monitors and handles all wireless links for satisfying end-to-end delay requirements and improving end-to-end throughput. The simulation results show that the proposed scheme can manage the trade-off between throughput and delay, which can improve end-to-end throughput without a delay constraint violation. In the future, we plan to consider general scenarios and other factors in wireless networks.

ACKNOWLEDGMENT

This research was funded by the MSIP (Ministry of Science, ICT and Future Planning), Korea in the ICT R&D Program 2015.

REFERENCES

[1] C. F. Chiasserini and M. Meo, “Modeling interactions between link layer and transport layer in wireless networks,” The 12th IEEE Int. Symp. PIMRC 2001, San Diego, USA, vol 1, Sep. 2001.

[2] F. Vaciraca, A. D. Vendictis, and A. Baiocchi,“Investigating interactions between ARQ mechanisms and TCP over wireless links,” in Proc. of European Wirel., Barcellona, Spain, Feb. 2004.

[3] A. Mehta, D. Kagaris, and R. Viswanathan,“Throughput performance of an adaptive ARQ scheme in rayleigh fading channels,” IEEE Trans. Wirel. Commun., vol. 5, no. 1, Jan. 2006.

[4] J. Han, B. Kim, and J. Lee. “TCP-friendly retransmission persistence management for SR-ARQ protocols,” IEICE Trans. Commun., vol. E92-B, no. 10, pp. 3243-3246, Oct. 2009.

[5] Software-Defined Networking: The New Norm for Networks" White paper. Open Networking Foundation. April 13, 2012

 ISBN: 978-1-61208-450-3